View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Analisis Harga Pokok Produksi Rumah Pada

RANCANG BANGUN APLIKASI MULTI-FACE DETECTOR MENGGUNAKAN METODE VIOLA JONES PADA FACE RECOGNITION

TUGAS AKHIR

Diajukan Sebagai Salah Satu Syarat Untuk Memperoleh Gelar Sarjana
Teknik Pada Jurusan Teknik Informatika

Oleh :

MUKSIT SYAHLAN MUHAIMIN

NIM. 10951006752

FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI SULTAN SYARIF KASIM RIAU
PEKANBARU
2013

LEMBAR PERSETUJUAN

RANCANG BANGUN APLIKASI MULTI-FACE DETECTOR MENGGUNAKAN METODE VIOLA JONES PADA FACE RECOGNITION

TUGAS AKHIR

Oleh:

MUKSIT SYAHLAN MUHAIMIN

1 0 9 5 1 0 0 6 7 5 2

Telah diperiksa dan disetujui sebagai laporan tugas akhir

Di Pekanbaru, pada tanggal 12 Juli 2013

Koordinator Tugas Akhir	Pembimbing
Reski Mai Candra, ST, M.Sc	**M. Safrizal, ST, M.Cs**
NIK. 130 510 032	NIK. 130 508 074

RANCANG BANGUN APLIKASI MULTI-FACE DETECTOR MENGGUNAKAN METODE VIOLA JONES PADA FACE RECOGNITION

MUKSIT SYAHLAN M.
NIM : 10951006752

Tanggal Sidang : 08 Juli 2013
Tanggal Wisuda : Oktober 2013

Jurusan Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Sultan Syarif Kasim Riau
Jl. Soebrantas No. 155 Pekanbaru

ABSTRAK

Pendeteksian wajah (*face detection*) merupakan sebuah topik yang banyak diteliti oleh kalangan ilmuan sebagai kajian khusus dalam pengolahan citra. Hal ini dikarenakan asumsi bahwa informasi tentang wajah seseorang dapat diekstraksi dari citra yang kemudian direpresentasikan sebagai identitas diri seseorang untuk media pengenalan wajah. Meskipun aplikasi mengenai topik ini telah diimplementasikan, namun pada dasarnya teknologi ini belum matang sehingga penelitian masih perlu terus dikembangkan untuk memperoleh hasil yang diinginkan. Tujuan penelitian ini adalah membangun sebuah aplikasi yang mampu melakukan pendeteksian dan pengenalan wajah manusia dengan kemampuan *plural* secara *real time*. Untuk proses pendeteksian wajah, metode yang diterapkan adalah metode *Viola Jones* dengan memanfaatkan sebuah *image processing library* yang berfungsi sebagai dasar pengolahan dalam pendeteksian citra wajah. Sedangkan untuk proses pengenalan wajah metode yang digunakan adalah *eigenface* berbasis *PCA* (*Principal Component Analysis*). Dalam penelitian ini, pengujian deteksi wajah maupun pengenalan wajah dilakukan berdasarkan beberapa faktor yang memungkinkan dapat mempengaruhi akurasi dari pendeteksian maupun pengenalan wajah. Faktor tersebut berupa pengaruh umur, gaya wajah, penambahan aksesoris, dan pendeteksian *plural*. Hasil pengujian menunjukkan perolehan tingkat akurasi pendeteksian wajah sebesar 75%, sedangkan tingkat akurasi pengenalan wajah sebesar 91%.

Kata Kunci : **Metode *eigenface*, Metode *viola jones*, *PCA* (*Principal Component Analysis*), *Pendeteksian wajah*, *Pengenalan wajah*, *Plural***

DESIGN APLICATION MULTI-FACE DETECTOR USING VIOLA JONES METHOD FOR FACE RECOGNITION

MUKSIT SYAHLAN M.
NIM : 10951006752

Date Of Final Exam : *08 July 2013*
Date Of Graduation : *October 2013*

Departement of Information
Faculty of Science and Technology
State Islamic University of Sultan Syarif Kasim Riau
Jl. Soebrantas No. 155 Pekanbaru

ABSTRACT

Face detection is a topic widely studied by the scientists as a special study in image processing. This is due to the assumption that information about a person's face can be extracted from the image which is then represented as a person's identity to the media of face recognition. Although applications about this topic have been implemented, but basically this technology is not yet mature to research so still needs to developed to obtain the desired results.The core of this research aims to build an application that is able to perform the detection and recognition of human faces with the plural ability in real time. For the face detection process, the method applied is Viola Jones by using an image processing library that serves as the basis for the detection of facial image processing. As for the face recognition process using the eigenface method that based PCA (Principal Component Analysis). In this research, testing of face detection and face recognition was based on several factors that could potentially affect the accuracy of detection and face recognition. The factors such as the influence of age, face style, the addition of accessories, and the plural detection. The testing results showed the acquisition of face detection accuracy rate of 75%, while the rate of the face recognition accuracy of 91%.

Keywords : ***Eigenface method, Face detection, Face recognition, PCA (Principal Component Analysis), Plural, Viola Jones method***

KATA PENGANTAR

بســـم الله الرحمن الرحـيم

السلام عليكم ورحمة الله وبركاته

Segala puji hanya milik Allah SWT yang telah melimpahkan ketenangan dan ketentraman dibatin yang terdalam. Shalawat dan salam semoga selalu tercurah kepada junjungan alam Nabi Muhammad SAW beserta keluarga, para sahabat serta pengikut setianya hingga akhir zaman. Amin

Alhamdulillahirobbil'alamin atas rahmat dan hidayah-Nya penulis dapat menyelesaikan laporan tugas akhir yang berjudul **"RANCANG BANGUN APLIKASI MULTI-FACE DETECTOR MENGGUNAKAN METODE VIOLA JONES PADA FACE RECOGNITION"** sebagai salah satu persyaratan akademik kelulusan pada jenjang sastra-1 Jurusan Teknik Informatika Universitas Islam Negeri Sultan Syarif Kasim Riau.

Selama pelaksanaan penelitian ini, penulis banyak mendapat pengetahuan, bimbingan, dukungan, dan arahan dari semua pihak yang telah membantu hingga penulisan laporan ini dapat terselesaikan. Untuk itu pada kesempatan ini penulis ingin menyampaikan ucapan terima kasih kepada :

1. ALLAH SWT, atas setiap ridho, cinta dan nikmat yang telah diberikan untuk setiap makhluk-Nya.
2. Rasulullah SAW, seorang pimpinan umat islam sekaligus panutan umat islam yang tiada henti-hentinya berjuang demi menyebarkan agama islam.
3. Terima kasih kepada Kedua orang Tua Penulis, Ibu dan Bapak yang tiada hentinya memanjatkan doa, memberikan dukungan dan semangat untuk kesuksesan penulis.
4. Bapak Prof. Dr. H. M. Nazir, selaku Rektor Universitas Islam Negeri Sultan Syarif Kasim Riau.
5. Ibu Dra. Hj. Yenita Morena, M.Si, selaku Dekan Fakultas Sains dan Teknologi Universitas Islam Negeri Sultan Syarif Kasim Riau.

6. Bapak Novriyanto, ST, M.Sc. selaku Ketua Jurusan Teknik Informatika Fakultas Sains dan Teknologi UIN SUSKA RIAU.
7. Bapak Reski Mai Candra ST, M.Cs. selaku Koordinator Tugas Akhir Jurusan Teknik Informatika Fakultas Sains dan Teknologi UIN SUSKA RIAU. Terima kasih atas semua saran yang telah Bapak berikan.
8. Bapak M. Safrizal, ST, M.Cs selaku pembimbing Tugas Akhir penulis. Terima kasih atas semangat, motivasi dan bimbingan yang diberikan kepada penulis selama pembuatan laporan Tugas Akhir. Semua nasihat, pesan, saran dan kritikan yang bapak berikan akan senantiasa penulis terapkan.
9. Nico Manggala yang telah banyak memberikan motivasi dan masukan serta ide-ide cemerlang kepada penulis dalam menyelesaikan Tugas Akhir ini.
10. Tentor Light Study Center Pekanbaru yang telah banyak membantu penulis dalam melaksanakan Tugas Akhir ini. Terima kasih atas perhatian dan semangat yang telah diberikan.
11. Alfi Syahri Lubis, Rio Fernando, Muhammad Faisal, Andika Widiyanto, Aditya Ayu Hapsari, Weni Rahim, Dian Permata Sari dan Aldila Dwi Nastiti serta semua teman-teman kelas TIF B angkatan 09 yang tidak mungkin disebutkan satu-persatu, yang telah banyak membantu dan memberi motivasi kepada penulis. Saya harap semuanya tetap semangat.
12. Semua pihak yang terlibat baik langsung maupun tidak langsung dalam pelaksanaan tugas akhir ini yang tidak dapat penulis sebutkan satu persatu.

Penulis menyadari bahwa dalam penulisan laporan ini masih banyak kesalahan dan kekurangan, oleh karena itu kritik dan saran yang sifatnya membangun sangat penulis harapkan untuk kesempurnaan laporan ini. Akhirnya penulis berharap semoga laporan ini dapat bermanfaat bagi penulis khususnya maupun pembaca umumnya. Akhir kata penulis ucapkan terima kasih dan selamat membaca.

Pekanbaru, 08 Juli 2013

Penulis

DAFTAR ISI

Halaman

BAB I
PENDAHULUAN

1.1. Latar Belakang

Wajah manusia merupakan kunci pembeda yang digunakan sebagai tanda pengenal atau sebagai identitas suatu personal (seseorang) dikarenakan memiliki keunikan atau ciri-ciri yang khusus yang berbeda. Salah satu permasalahan yang berkaitan dengan wajah adalah pendeteksian wajah (*face detection*), dimana permasalahan ini telah begitu populer dikalangan para ilmuan, dan sebagai bidang kajian khusus didalam pengolahan citra. *Face detection* telah mencakupi berbagai penerapan bidang aplikasi seperti *interfacing* manusia dengan komputer, *surveillance system*, *content based image retrieval*, *video conferencing*, *financial transaction*, *forensic application*, *pedestrian detection*, *image database management system* dan lain-lain.

Pendeteksian wajah (*face detection*) adalah salah satu tahap awal yang sangat penting dalam sistem pengenalan wajah (*face recognition*) yang digunakan dalam identifikasi biometrik, sistem ini dapat diterapkan dalam berbagai disiplin ilmu, antara lain: bidang psikologi, teknik dan *Neuroscience*. Deteksi wajah juga dapat digunakan untuk pencarian atau pengindeksian data wajah dari citra atau *video* yang berisi wajah dengan berbagai ukuran, posisi, dan latar belakang. Pendeteksian wajah (*face detection*) secara otomatis dengan bantuan komputer merupakan permasalahan yang tidak mudah karena wajah manusia memiliki tingkat variabilitas yang tinggi baik variabilitas *intra-personal* maupun *extra-personal*. Sehingga membutuhkan algoritma yang kompleks untuk merepresentasikan wajah manusia kedalam sebuah sistem komputer *vision*. Terdapat berbagai macam teknik penyelesaian yang digunakan dalam melakukan penelitian tersebut, diantaranya adalah Algoritma *Quickprop*, teknik *Active Learning*, Metode *Rowley*, dan lain-lain. Dalam penelitan ini teknik yang digunakan untuk pendekteksian wajah adalah menggunakan metode *Viola Jones* dimana metode ini digunakan untuk mengenal wajah secara cepat dan akurat

dalam mendeteksi atau menunjukkan bagian citra yang terdapat atau mengandung wajah didalamnya dengan memanfaatkan sebuah *image processing library*.

Algoritma yang diterapkan dalam *Viola Jones* menggunakan sebuah tipe *face detector* yang disebut *Haar-cascade classifier*. Jika ada sebuah citra (bisa dari file *live* video), *face detector* akan menguji tiap lokasi citra dan mengklasifikasinya sebagai "wajah" atau "bukan wajah". Klasifikasi wajah ini menggunakan sebuah pemisalan skala yang tetap, misalnya 20 × 20 *pixel*. Jika wajah pada citra lebih besar atau lebih kecil dari *pixel* tersebut, *classifier* terus menerus jalan beberapa kali, untuk mencari wajah pada gambar tersebut.

Sistem pendeteksian wajah manusia dapat dilakukan berbagai macam pengembangan disiplin ilmu sesuai kebutuhan dan bidang penelitian. Dalam penelitian ini proses pendeteksian wajah akan diteruskan pada tahap pengenalan wajah (*face recognition*). Pengenalan wajah manusia dalam gambaran visual dapat di-implementasikan ke dalam banyak aplikasi dengan menerapkan beragam metode, hal ini bergantung urgensi dan implementasi pada bidang yang dikaji. Banyak metode yang dapat diimplemetasikan dalam sistem pengenalan wajah ini, salah satunya adalah metode *eigenface* berbasis PCA (*Principal Component Analysis*), dimana citra yang telah diproses direduksi untuk menghasilkan vektor berbasis *orthogonal* yang disebut *eigenface*.

Teknologi pengenalan wajah menggunakan metode *eigenface* berbasis PCA saat ini telah banyak diteliti, diantaranya adalah: Puspitodjati, dkk. (2012) tentang sebuah sistem yang mampu mengidentifikasi emosi dengan berbagai ekspresi seseorang dengan menerapkan Metode *Eigenface* dan *Nearest Feature Line*, "Sistem Pengenalan Ekspresi Wajah Berdasarkan Citra Wajah Menggunakan Metode *Eigenface* dan *Nearest Feature Line"*, kemudian Bayu, dkk. (2012) tentang pengenalan wajah dalam sistem keamanan rumah "Penerapan *Face Recognition* dengan Metode *Eigenface* dalam *Inteligent Home Security*", kemudian Lata, dkk. (2009) tentang pengenalan wajah dengan menerapkan metode *eigenface* dan *PCA* "*Facial Face Recognition Using Eigenface By PCA*", dan Zayuman (2008), tentang pengenalan wajah dengan menggunakan metode

PCA dan *Back Propagation* "Pengenalan Wajah Manusia Menggunakan Analisis Komponen Utama (PCA) dan Jaringan Syaraf Tiruan Perambatan-Balik".

Dengan menggunakan aplikasi pendeteksi dan pengenalan wajah atau *face recognition* diharapkan dapat dikembangkan kedalam berbagai sistem yang membutuhkan aspek pengenalan wajah. Ada banyak pengembangan sistem yang menerapkan *face recognition* ini seperti bidang *intelligent security* dan *intelligent service*. Inilah yang menjadi alasan yang melatarbelakangi penulis untuk melakukan penelitian ini, yaitu membuat sebuah aplikasi pengenalan wajah yang mampu mendeteksi dan mengenali wajah manusia (*face recognition*) secara cepat dan akurat.

1.2. Rumusan Masalah

Berdasarkan penjelasan yang telah diuraikan dibagian latar belakang di atas, maka dapat ditarik sebuah rumusan masalah yang akan dijelaskan lebih lanjut dalam laporan tugas akhir ini, yaitu bagaimana merancang serta membangun sebuah aplikasi *multi-face detector* menggunakan metode *Viola Jones* pada *face recognition.*

1.3. Batasan Masalah

Agar tidak terjadi kesalahan persepsi dalam memahami kajian yang terdapat dalam laporan tugas akhir ini, maka berikut dijelaskan beberapa hal yang menjadi batasan masalah laporan ini:

1. Metode yang digunakan untuk *training* dalam mereduksi dimensi citra wajah adalah menggunakan metode *eigenface* berbasis PCA (*Prinsipal Component Analysis*).
2. *Face detector* untuk proses *multi-face recognition* menerapkan *Image Processing Library OpenCV* berdasarkan pada metode *Viola Jones*.
3. Citra wajah yang diambil dalam pendeteksian wajah adalah pada posisi depan (*frontal face*).

4. Penghitungan *Similarity Distance Measure* adalah menggunakan algoritma *Eucledian Distance.*
5. Dalam penelitian tidak dibahas metode khusus untuk menangani kemungkinan hal-hal yang dapat mempengaruhi citra hasil *capturing image* untuk pendeteksian wajah maupun pengenalan wajah, seperti: *brightness* atau pencahayaan, perubahan skala, perubahan posisi, perubahan detil dan ekspresi pada wajah.

1.4. Tujuan Penelitian

Tujuan yang ingin dicapai pada penelitian terhadap kasus yang dibahas dalam laporan ini adalah untuk merancang serta membangun sebuah aplikasi *multi-face detector* menggunakan metode *Viola Jones* pada *face recognition* berdasarkan metode *eigenface* berbasis *PCA* (*Principal component Alalysis*). Serta menunjukkan seberapa akuratkah implementasi dari metode tersebut untuk menagkap serta mengenali citra wajah manusia.

1.5. Sistematika Penulisan

Berikut merupakan rencana susunan sistematika penulisan laporan tugas akhir yang akan dibuat. Sistematika penulisan laporan tugas akhir ini meliputi:

1. **Bab I Pendahuluan**

   Bab I ini merupakan bagian yang akan menguraikan hal-hal seperti; latar belakang masalah, rumusan masalah, batasan masalah, tujuan penelitian, dan sistematika penulisan laporan tugas akhir
2. **Bab II Landasan Teori**

   Bab ini berisi tentang teori-teori yang berkaitan *multi-face detector* menggunakan metode *Viola Jones* pada *face recognition*
3. **Bab III Metodologi Penelitian**

   Bab ini berisi tentang cara-cara atau hal-hal yang dilakukan dalam menyelesaikan kasus tugas akhir ini.

4. **Bab IV Analisa dan Perancangan**

   Bab ini berisi tentang analisa dari penelitian yang dilakukan dalam tugas akhir ini sekaligus menerangkan perancangan aplikasi yang dibangun yaitu aplikasi *multi-face detector* menggunakan metode *Viola Jones* pada *face recognition*

5. **Bab V Implementasi dan Pengujian**

   Bab ini berisi tentang langkah-langkah perancangan aplikasi yaitu aplikasi *multi-face detector* menggunakan metode *Viola Jones* pada *face recognition*

6. **Bab VI Penutup**

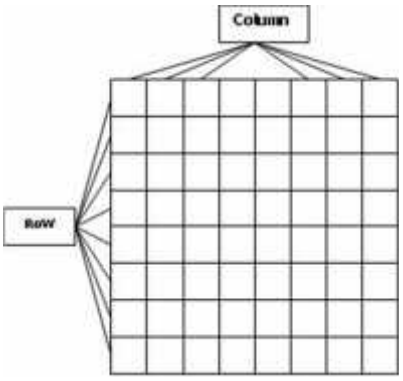
   Bab ini berisi kesimpulan dan saran mengenai hasil analisa, perancangan, hasil implementasi dan hasil pengujian yang telah dilakukan terhadap rancang bangun aplikasi *multi-face detector* menggunakan metode *Viola Jones* pada *face recognition*.

BAB II
LANDASAN TEORI

2.1. Citra Digital

Pengertian citra secara umum adalah merupakan suatu gambar, foto ataupun berbagai tampilan dua dimensi yang menggambarkan suatu visualisasi objek. Citra dapat diwujudkan dalam bentuk tercetak ataupun digital. Citra digital adalah larik angka-angka secara dua dimensional (Liu and Mason, 2009). Citra digital tersimpan dalam suatu bentuk larik (*array*) angka digital yang merupakan hasil kuantifikasi dari tingkat kecerahan masing-masing piksel penyusun citra tersebut.

Citra digital yang tersimpan dalam larik dua dimensi tersusun atas unsur-unsur kecil yang disebut dengan piksel. Masing-masing piksel terkait secara spasial dengan area di permukaan bumi. Struktur *array* ini tersusun dalam baris horisontal yang disebut baris (*Lines*) dan kolom vertikal (*Samples*). Masing-masing piksel dalam *raster* citra menyimpan nilai tingkat kecerahan piksel yang direpresentasikan sebagai suatu angka digital. Susunan piksel dalam struktur *array* citra digital yang tersebut disebut dengan data *raster*. Lebih Jelasnya dapat dilihat pada gambar 2.1 dibawah ini.



Gambar 2.1 Kolom dan Baris Membentuk Piksel Data *Raster*

Sebagai suatu susunan dari angka digital, beberapa bentuk operasi matematis dapat diberlakukan terhadap citra digital tersebut. Operasi matematis atas suatu citra digital disebut dengan pengolahan citra digital.

2.2. Pengolahan Citra Digital

Pengolahan citra merupakan suatu proses yang dilakukan dengan input berupa citra dan output-nya pun berupa citra. Proses ini dilakukan pada citra data *training* sebelum *feature extraction* (pengambilan ciri). Terdapat dua hal yang mendasar dalam memahami proses pembentukan citra, yaitu:

1. Geometri formasi citra yang menentukan lokasi suatu titik dalam pemandangan yang diproyeksikan pada bidang citra
2. Fisik cahaya yang menentukan *Brighness* suatu *pixel* citra sebagai fungsi pencahayaan

Oleh karena itu, diperlukan penghubung notasi matematika untuk mengembangkan algoritma pengolahan citra dan notasi algoritma yang digunakan untuk membuat program komputer yang disimpan kedalam sistem penyimpanan memori dua dimensi yang disebut larik (*array*).

Tujuan utama dari pengolahan citra adalah untuk perbaikan data citra dengan menekan *noise* yang tidak diinginkan didalam citra sehingga dapat diproses dan dianalisa lebih lanjut. Pada proses ini citra dinormalisasi agar lebih siap untuk proses selanjutnya. Proses ini memiliki tahapan-tahapan tertentu dengan berbagai metode dan kalkulasi tertentu sesuai kebutuhan penelitian.

Dalam proses pendeteksian wajah yang kemudian dilanjutkan pada proses pengenalan wajah, terdapat serangkaian proses pengolahan citra untuk membuat ruang citra lebih sederhana agar mudah diproses ketahap selanjutnya, proses ini dinamakan normalisasi citra. Pengolahan normalisasi citra secara garis besar terdapat dua proses, yaitu:

1. RGB to *Grayscale*

   Merupakan proses konversi warna dari citra RGB (*Red, Green, Blue*) menjadi citra *Grayscale* (Keabuan). Sederhananya, *grayscale* pada sebuah *digital image* adalah citra yang setiap *pixel*nya berisikan

informasi intensitas warna hitam atau putih. *Grayscale* lebih mudah diproses karena mengandung nilai lebih sedikit, yaitu 8 bit warna dibanding *RGB* yang mengandung 24 bit warna. Berikut persamaan yang digunakan untuk merubah citra RGB menjadi *Grayscale:*

Grayscale = (R*0,2126) + (G*0,7152) + (B*0,0722)……...2.1

Keterangan : R = Warna merah, G = Warna hijau dan B = Warna biru

Banyak metode yang dapat digunakan dalam mengkonversi citra *RGB* menjadi *Grayscale,* namun penulis dalam penelitian ini menggunakan metode *Luminance.* Metode ini banyak dipakai dalam perangkat lunak karena lebih menitik beratkan pada nilai hijau dengan anggapan manusia lebih cenderung lebih sesnsitif dengan warna hijau dari pada warna lainnya.

2. *Dimension Reduction*

   Merupakan proses untuk mengubah dimensi citra dari dimensi yang berjumlah M menjadi N dimana N < M. Proses ini bertujuan untuk memperkecil ukuran citra yang diolah sehingga dapat mempercepat proses selanjutnya. Hasil dari reduksi ini berupa *matrix* kolom yang kemudian dimasukan kedalam matriks augmentasi (Alfin Soleh, 2013).

2.3. *Face Detection* Dengan Metode *Viola-Jones*

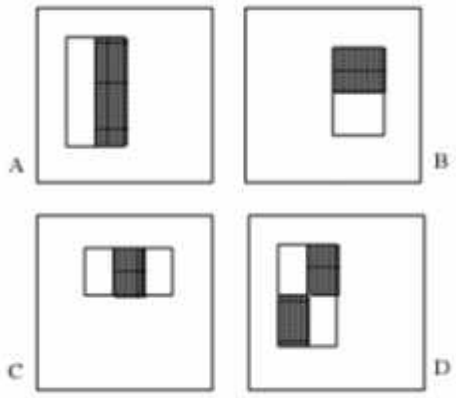
Pendeteksian objek merupakan salah satu topik dalam visi komputer yang cukup banyak dipelajari dan dikembangkan saat ini, baik oleh para pelajar, maupun para ahli. Salah satu metode pendeteksian objek yang cukup populer adalah metode *Viola Jones*, yang diusulkan oleh Paul Viola dan Michael Jones pada tahun 2001. Metode ini merupakan metode pendeteksian objek yang mampu memberikan hasil dengan tingkat keakuratan yang cukup tinggi, dan dengan kecepatan yang sangat tinggi.

Dalam penelitiannya, Viola dan Jones menggunakan 4916 citra berisi wajah dan 10.000 *sub-window* yang tidak berisi wajah untuk proses *training*. Setiap inputan kemudian dihitung nilai fiturnya menggunakan citra integral. Hasil perhitungan di-*training* menggunakan suatu algoritma *boosting* yang merupakan

variasi dari algoritma *AdaBoost*. Hasil *training* inilah yang digunakan untuk membentuk *cascaded classifier*, yang digunakan untuk mengklasifikasikan wajah. Terdapat empat kunci utama di dalam metode pendeteksian objek *Viola Jones*, yaitu:

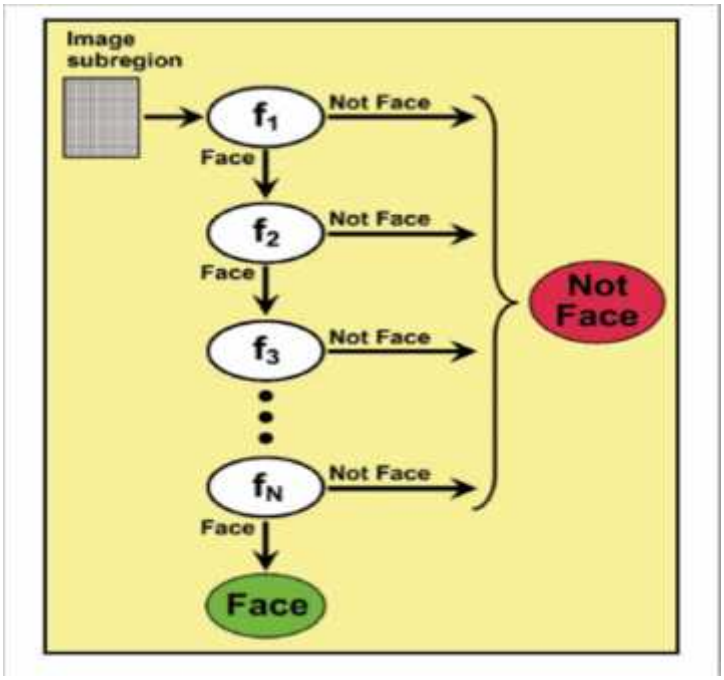
1. Fitur segi empat sederhana disebut sebagai *fitur Haar*.
2. Integral *image* (citra integral) untuk pendeteksian fitur secara cepat.
3. Metoda *machine learning* AdaBoost (algoritma *boosting*).
4. *Cascade Classifier* (Klasifier bertingkat) untuk menghubungkan dan mengelompokkan banyak fitur secara efisien.

Fitur yang digunakan oleh Viola dan Jones didasarkan pada *Wavelet Haar*. *Wavelet Haar* adalah gelombang tunggal bujur sangkar (satu interval tinggi dan satu interval rendah). Untuk dua dimensi, satu terang dan satu gelap. Kombinasi-kombinasi yang digunakan untuk pendeteksian objek visual tidak terlalu menyerupai *Wavelet Haar* yang sebenarnya. Walaupun demikian, kombinasi-kombinasi segi empat itu cocok untuk tugas-tugas pengenalan visual yang lebih baik. Oleh karena itu fitur ini disebut fitur *Haar*, atau fitur *Haarlike*, bukan *Wavelet Haar*. Gambar 2.2 dibawah ini menunjukkan fitur yang digunakan dalam *OpenCV* (Kurniawan, agus, dkk. 2009).



Gambar 2.2 Contoh Gambar Fitur dalam Metode *Viola-Jones*

Untuk memilih fitur *Haar* yang digunakan dan untuk mengubah nilai *threshold*, Viola dan Jones menggunakan metode *machine-learning* yang disebut *AdaBoost*. *AdaBoost* menggabungkan banyak *classifier* untuk membuat satu *classifier*. Masing-masing *classifier* menetapkan suatu bobot, dan gabungan dari bobot inilah yang akan membentuk satu *classifier* yang kuat. Viola dan Jones menggabungkan serangkaian *AdaBoost classifier* sebagai rantai *filter* atau *filter chain*. Masing-masing filter merupakan *AdaBoost classifier* yang terpisah dengan jumlah *weak classifier* yang sedikit dan sama.



Gambar 2.3 *Classifier Cascade Image*

Berdasarkan gambar 2.3 diatas dapat dilihat bahwa filter pada masing-masing level dilatih untuk mengklasifikasikan gambar yang sebelumnya telah di-*filter* (*Training set* merupakan *database* dari wajah). Selama penggunaannya, jika satu dari *filter-filter* tersebut gagal, *image region* atau daerah pada gambar diklasifikasikan sebagai "Bukan Wajah". Saat *filter* berhasil melewatkan *image region*, *image region* kemudian masuk pada *filter* yang selanjutnya. *Image region* yang telah melalui semua *filter* akan dianggap sebagai "Wajah" (Viola, P; Michael J; Jones, 2001).

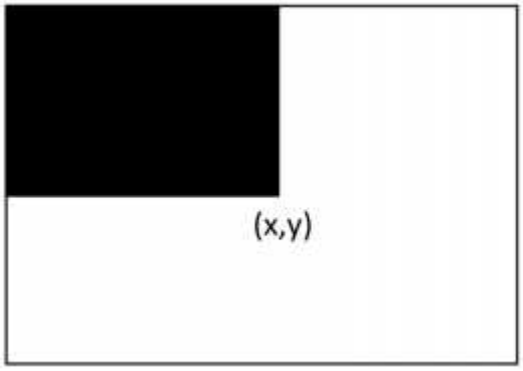
2.3.1. Fitur *Haar* (*Haarlike Feature*)

Fitur merupakan tahap paling awal yang diperlukan dalam pendeteksian objek dengan menggunakan metode *Viola Jones*. Penggunaan fitur dilakukan karena pemrosesan fitur berlansung lebih cepat dibandingkan dengan pemrosesan *image per pixel*.

Fitur *haar* (*haarlike feature*) adalah pengambilan *pixel* pada citra membentuk segi empat sederhana yang dibagi menjadi dua bagian, satu bagian gelap dan satu bagian terang. Adanya fitur *haar* ditentukan dengan cara mengurangi rata-rata *pixel* pada daerah gelap dari rata-rata *pixel* dari daerah terang. Jika nilai perbedaan itu diatas nilai ambang atau threshold, maka dapat dikatakan bahwa fitur itu ada. Untuk menentukan ada tidaknya dari ratusan fitur haar pada sebuah citra dan pada skala yang berbeda secara efisien, *Viola Jones* menggunakan teknik yang dinamakan citra integral atau *integral image* (Kurniawan, agus, dkk. 2009).

2.3.2. Citra Integral

Citra integral merupakan suatu representasi citra baru, dimana nilai piksel dari suatu titik (x,y) pada citra merupakan hasil penjumlahan dari seluruh piksel yang ada disebelah kiri dan atas titik tersebut. Citra integral sangat membantu dalam perhitungan fitur *Haarlike*. Dengan menggunakan citra integral, perhitungan fitur *Haarlike* dapat dilakukan dengan sangat cepat.



Gambar 2.4 Nilai Piksel pada Titik (x,y)

Berdasarkan gambar 2.4 diatas, nilai (x,y) adalah nilai dari seluruh piksel pada daerah yang diarsir. Untuk dapat lebih memahami bagaimana simulasi perhitungan dari citra integral ini, maka dapat dilihat bab 4 tentang pemrosesan citra integral pada proses pendeteksian wajah.

Citra Asli

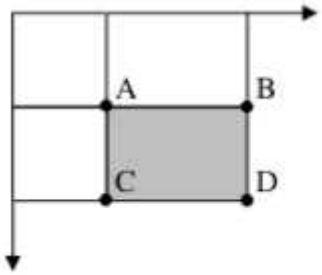
3	6	9	8	7
9	2	7	1	9
5	1	8	7	4
2	4	2	5	9
7	6	2	1	4

Citra Setelah di-integral-kan

3	9	18	26	33
12	20	36	45	61
17	26	50	66	86
19	32	58	79	108
26	45	73	95	128

Gambar 2.5 Perbandingan Citra Asli dengan Citra integral

Gambar 2.5 diatas menggambarkan perbedaan antara nilai piksel citra asli dengan citra setelah dilakukan proses peng-integral-an. Dengan mendapatkan nilai dari citra integral maka jumlah dari seluruh piksel yang ada dalam setiap persegi panjang dapat dihitung dengan empat nilai. Nilai-nilai ini merupakan piksel pada citra integral yang bertepatan dengan sudut-sudut persegi panjang yang ada pada citra masukkan.

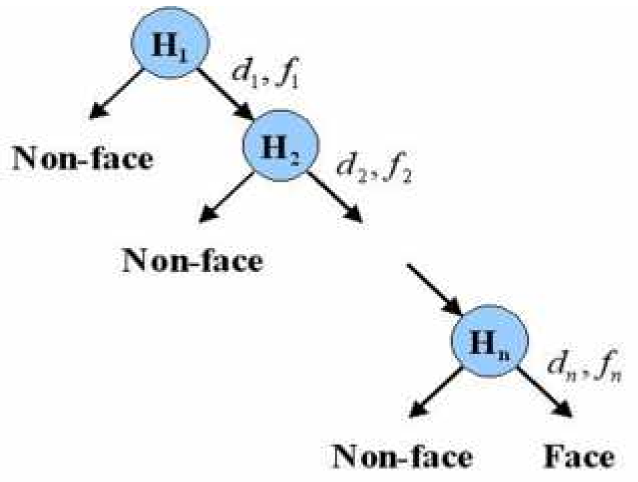


ABCD = D - (B + C) + A

Gambar 2.6 Jumlah seluruh Piksel yang ada pada Persegi Panjang

2.3.3. Algoritma *Boosting*

Viola Jones menggunakan algoritma *boosting* yang merupakan variasi dari algoritma *AdaBoost*. Tujuan dari algoritma *boosting* adalah untuk membentuk suatu *template* objek yang akan dideteksi, dalam hal ini adalah wajah. Secara tidak langsung algoritma *AdaBoost* ini dapat dikatakan sebagai rantai *filter*, seperti yang ditunjukkan pada Gambar 2.7. Setiap filter merupakan *classifier AdaBoost* terpisah dengan jumlah *weak classifiers* yang relatif kecil.



Gambar 2.7 Algoritma *AdaBoost*

Untuk lebih jelasnya dapat dilihat pada langkah-langkah algoritma dibawah ini:

Diberikan citra: (x_1, y_1),…. (x_n, y_n)
di mana $y_i = 0,1$ untuk data negatif dan positif berturut-turut.

Untuk setiap citra *training*, diberi koordinat (x,y) dengan $y = 0$ untuk citra yang tidak mengandung wajah (citra negatif), dan $y = 1$ untuk citra berisi wajah (citra positif).

Inisialisasikan bobot:

$W_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ ………………………………………………………….. 2.2

untuk $y_i = 0,1$ berturut-turut, di mana m dan l adalah jumlah negatif dan positif berturut-turut.

Setiap citra diberi bobot awal yang sama, $\frac{1}{2m}$ untuk citra negatif, dan $\frac{1}{2l}$ untuk citra positif. Di mana m adalah jumlah total citra negatif, dan l adalah jumlah total citra positif yang digunakan dalam proses *training.*

Untuk $t = 1,\ldots,T$:

1. Normalisasikan bobot,

$W_{t,i} \leftarrow \frac{wt,i}{\sum_{j=1}^{n} wt,j}$ …….. 2.3

Sehingga W_t adalah distribusi probabilitas.

2. Untuk setiap fitur *j* latih sebuah *classifier* yang dibatasi agar menggunakan sebuah fitur tunggal. Kesalahan dievaluasi dilakukan dengan persamaan:

$W_t, \epsilon_j = \sum_i w_I |h_j(x_i) - y_i|$ …….. 2.4

3. Pilih *classifier* h_t, dengan kesalahan terendah ϵ_t
4. Perbaharui bobot:

$W_{t+1,i} = W_{t,i} \beta_t^{1-e^i}$ ……. 2.5

di mana $e_i = 0$ jika data x_i diklasifikasi dengan benar, $e_i = 1$ jika sebaliknya,

dan $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$ …….. 2.6

Untuk setiap tahapan t:

1. Normalisasikan bobot untuk mendapatkan distribusi probabilitas atau kandidat *classifier* lemah.
2. Evaluasi setiap kandidat *classifier* lemah tersebut.
3. Pilih kandidat *classifier* lemah dengan kesalahan yang paling sedikit, tetapkan sebagai *classifier* lemah.
4. Klasifikasi semua data *training* menggunakan *classifier* lemah yang telah didapatkan, dan lakukan pemberian bobot ulang terhadap data-data tersebut. Perbesar bobot semua data yang mengalami kesalahan klasifikasi, dan kurangi bobot (kembalikan ke bobot awal) semua data yang telah diklasifikasi dengan benar. Hal ini ditujukan agar setiap kesalahan klasifikasi yang terjadi dapat terlihat dan diatasi oleh *weak classifier* yang terpilih pada tahapan selanjutnya.

Proses *looping* terjadi pada algoritma di atas, hingga data-data dapat diklasifikasi dengan benar.

Classifier kuat akhir-nya adalah:

$h(x)=\{^{1}_{0}\ \sum_{t=1}^{T} a_t h_t(x) \geq \frac{1}{2}\sum_{t=1}^{T} a_t$ 2.7

dimana $a_t = log\ \frac{1}{\beta_t}$

Classifier akhir yang didapatkan merupakan gabungan dari semua *classifier* lemah yang didapatkan dari setiap tahapan *boosting*.

Berdasarkan langkah-langkah algoritma diatas, simulasi perhitungan *AdaBoost* yang secara umum dipakai untuk pengklasifikasian ialah sebagai berikut :

1. Misalkan terdapat 30 gambar untuk training fitur dimana diantara gambar tersebut terdapat 6 gambar positif dan selebihnya adalah gambar negatif.
2. *Input* berupa : gambar untuk *training* (x_i, y_i), dimana x_i adalah nomor gambar ke-*i*, y_i = 1 untuk gambar positif, y_i = 0 untuk gambar negatif, dan *i* ialah jumlah gambar dengan i = 1.....N.
   Contoh : gambar ke-1,..., ke-30 → (1,1),...,(6,1),(7,0),....,(30,0)
2. Inisialisasi : Bobot awal berdasarkan persamaan 2.2 dengan m = jumlah gambar negatif, dan *l* = jumlah gambar positif.
   Contoh : m = 24, *l* = 6 maka, $W_{1,1} = \frac{1}{2x6}$ = 0,0833, $W_{1,30} = \frac{1}{2x24}$ = 0,0208
3. For t = 1.....T , dimana t ialah jumlah iterasi (tergantung dari banyaknya fitur yang akan dipilih)
   a. Normalisasi bobot, sehingga penjumlahan total dari semua bobot ialah satu. w_t adalah bobot gambar pada iterasi ke-t. Proses ini berdasarkan pada persamaan 2.3.
   b. Untuk setiap fitur, j, latih *classifier* h_j. Evaluasi *error rate* atau kesalahan, proses ini berdasarkan pada persamaan 2.4.
   c. Pilih Fitur dengan *error rate* terkecil ($\in_t$)

d. *Update* atau perbaharui bobot berdasarkan persamaan 2.5, yaitu dengan e_i = 1 bila gambar diklasifikasi dengan benar dan 0 bila diklasifikasi salah, dimana $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$

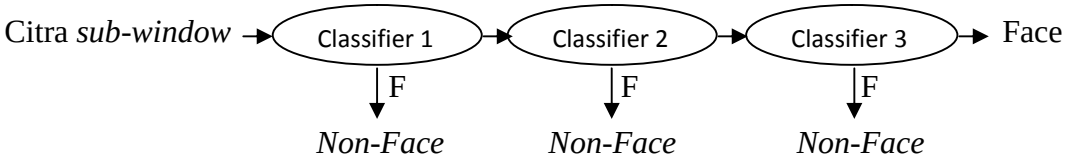
4. Hasil akhir klasifikasi yang bagus adalah diperoleh *strong classifier* dimana $a_t = log \frac{1}{\beta_t}$ proses ini berdasarkan persamaan 2.7.

2.3.4. *Cascaded Classifier*

Cascaded classifier merupakan suatu metode pengklasifikasian bertingkat, dimana *input* dari setiap tingkatan merupakan *output* dari tingkatan sebelumnya. Pada *classifier* tingkat pertama, yang menjadi inputan adalah seluruh citra *sub-window*. Semua citra *sub-window* yang berhasil melewati *classifier* pertama akan dilanjutkan ke *classifer* ke dua, dan seterusnya. Apabila suatu *sub-window* berhasil melewati semua tingkat *classifier*, maka *sub-window* tersebut dinyatakan sebagai wajah. Sedangkan untuk *sub-window* yang gagal melewati suatu tingkat *classifier* akan langsung dieliminasi dan dinyatakan sebagai bukan wajah (tidak akan diproses lagi). Hal ini sangat mempercepat proses pengklasifikasian, karena jumlah inputan yang diterima di setiap c*lassifier* akan semakin berkurang.

Cascaded classifier dirancang sedemikian rupa untuk meningkatkan tingkat pendeteksian dan mengurangi jumlah positif palsu. Setiap tingkatan *classifier* merupakan representasi hasil dari algoritma *boosting*. Jadi, di setiap tingkat *classifer* memiliki sejumlah *weak classifiers*. Setiap *weak classifier* memberikan aturan pasti mengenai fitur *Haarlike* yang digunakan (jenis, ukuran, dan lokasi), nilai *threshold* terbaik untuk setiap fitur, serta nilai batasan setiap fitur tersebut. Biasanya, semakin tinggi tingkat *classifer*, semakin banyak pula jumlah *weak classifier* yang ada. Hal ini mengakibatkan semakin sulitnya suatu *sub-window* untuk berhasil melewati tingkatan *classifier* tersebut, sehingga jumlah sub-*window* yang dieliminasi akan semakin banyak, dan jumlah *sub-window* yang berhasil lolos ke *classifier* tingkat selanjutnya akan semakin sedikit.

Oleh karena semakin sedikit *sub-window* yang berhasil lolos ke *classifier* selanjutnya, maka semakin sedikit pula jumlah *false positive* (citra negatif yang dianggap sebagai citra positif) yang berhasil lolos. Dengan berkurangnya *false positive*, tingkat keakuratan pendeteksian pun meningkat. Jadi, semakin banyak tingkat *classifier* di dalam suatu *cascaded classifier*, maka semakin akurat hasil yang akan didapatkan. Untuk lebih jelasnya proses filter dari masing-masing *classifier* yang dilalui setiap citra *sub-window* dapat dilihat pada gambar 2.7 berikut ini.



Gambar 2.8 Tahapan *Cascaded Classifier*

2.4. Pendeteksian Objek pada OpenCV

Metode pendeteksian objek pada *OpenCV* merupakan metode pendeteksian objek *Viola Jones*, yang kemudian dikembangkan oleh Lienhart. Fungsi untuk mendeteksi objek pada *OpenCV*, yaitu *cvHaarDetectObjects*. Untuk dapat menjalankan fungsi ini dibutuhkan suatu *cascaded classifier* yang berisi hasil *training* terhadap sejumlah data-data berupa citra positif (berisi wajah) dan citra negatif. Hasil *training* ini sebenarnya merupakan *template* dari bentuk objek yang akan dideteksi, misalnya wajah, mata, hidung, mulut, dan tubuh. *Cascaded classifier* ini telah disediakan oleh *OpenCV* dalam bentuk file .xml, sehingga dapat langsung dimasukkan ke dalam program menggunakan fungsi *cvLoad*. Untuk *template* wajah, *OpenCV* menyediakan lima jenis *template*, yaitu: *haarcascade_frontalface_alt.xml*, *haarcascade_frontalface_alt2.xml*, *haarcascade_frontalface_alt_tree.xml*, *haarcascade_frontalface_default.xml*, dan *haarcascade_profileface.xml*. Setiap *template* ini memiliki isi yang berbeda-beda, baik dari jumlah *stages*, jumlah *tree*, nilai *threshold*, serta bentuk, ukuran, dan posisi fitur *Haarlike* yang digunakan.

Berikut adalah potongan *code* sederhana dalam mendeteksi wajah menggunakan *haarcascade frontalface*.

```
CvHaarClassifierCascade *cascade =
(CvHaarClassifierCascade*)cvLoad("haarcascade_frontalface_alt.xml",0,0,
0);
CvMemStorage *storage = cvCreateMemStorage(0);
cvClearMemStorage(storage);

CvSeq* faces =
cvHaarDetectObjects(img,cascade,storage,1.1,2,CV_HAAR_DO_CANNY_PRUNING,
cvSize(24,24));
```

Gambar 2.9 Potongan Kode Program Pendeteksian Wajah

Dari potongan *code* berdasarkan gambar 2.8 di atas, dapat dilihat bahwa *template* wajah yang digunakan adalah *haarcascade_frontalface_alt.xml*. *Template* wajah ini akan dijelaskan lebih detil pada bagian 2.5. Dalam *cvHaarDetectObjects* memiliki tujuh *variables*, yaitu:

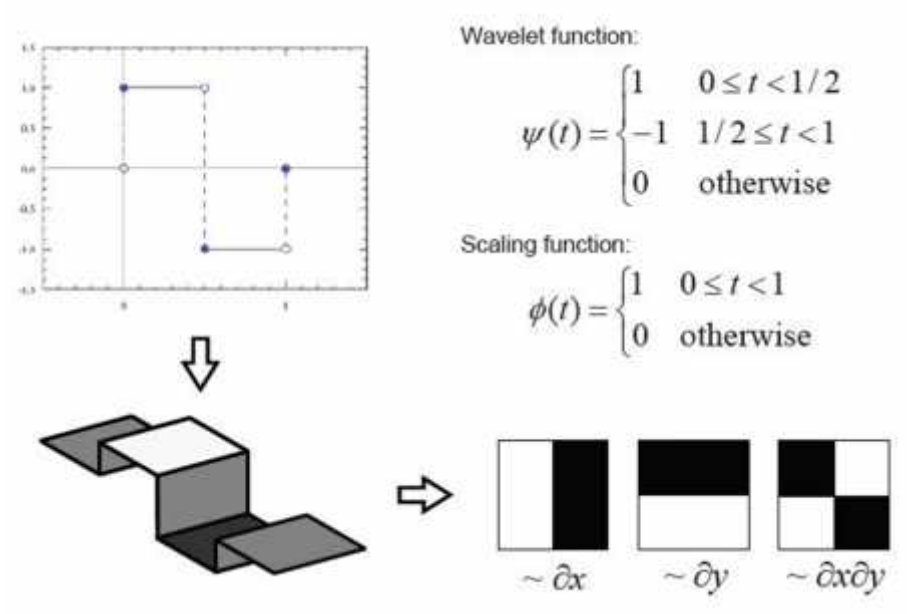
1. *const CvArr *img* yaitu citra yang akan dideteksi.
2. *CvHaarClassifierCascade *cascade* yaitu *cascaded classifier* yang digunakan.
3. *CvMemStorage *storage* yaitu memori penyimpanan asosiatif untuk menyimpan semua data yang diperlukan dalam proses klasifikasi, sehingga disebut juga sebagai *work buffer*.
4. *double scale_factor* yaitu faktor pembesaran untuk *sub-window*. Misalnya, jika *scale_factor* = 1.1, maka *sub-window* akan mengalami pembesaran sebesar 1.1 kali ukuran sebelumnya. *Sub-window* akan menelusuri setiap bagian dari citra, dimulai dari posisi (0,0) hingga posisi ujung kanan bawah citra. Setelah itu, *sub-window* akan kembali ke posisi awal (0,0) dengan ukuran yang lebih besar. Demikian seterusnya hingga ukuran sub-*window* sebesar ukuran citra yang dideteksi.
5. *int min_neighbors* yaitu jumlah minimal persegi di sekitar untuk membentuk suatu objek. Persegi di sini dimaksudkan sebagai tanda yang diberikan oleh detektor ketika suatu daerah citra berhasil melewati seluruh tingkatan *classifier*. *Default*-nya adalah 3.
6. *int flags* = 0 yaitu mode operasi yang digunakan. Angka 0 menunjukkan bahwa mode operasi yang digunakan adalah

CV_HAAR_DO_CANNY_PRUNING, yang berarti bahwa fungsi menggunakan *Canny edge detector* untuk menolak semua wilayah dalam citra yang mengandung terlalu sedikit ataupun terlalu banyak *edges*, sehingga tidak memungkinkannya mengandung objek yang dicari (dalam hal ini adalah wajah). Untuk *OpenCV* versi 1.1 dan versi sebelumnya, mode operasi yang ada hanya satu, yaitu *CV_HAAR_DO_CANNY_PRUNING*. Sedangkan untuk versi yang lebih baru, terdapat tiga mode tambahan, yaitu *CV_HAAR_SCALE_IMAGE*, *CV_HAAR_FIND_BIGGEST_OBJECT*, dan *CV_HAAR_DO_ROUGH_SEARCH*. Pada mode *CV_HAAR_SCALE_IMAGE*, yang mengalami pembesaran (*scaled*) adalah citra, bukan detektornya. Mode ini membutuhkan penggunaan memori dan *cache* yang lebih sedikit. *CV_HAAR_FIND_BIGGEST_OBJECT* digunakan untuk mendeteksi hanya satu objek dengan ukuran terbesar, dan *CV_HAAR_DO_ROUGH_SEARCH*, digunakan untuk melakukan pencarian yang kasar, di mana proses pencarian akan langsung dihentikan begitu kandidat pertama didapatkan. Setiap mode ini dapat digabungkan dengan mode lainnya menggunakan *operator* OR. Namun, untuk *CV_HAAR_DO_ROUGH_SEARCH* hanya dapat digabung dengan *CV_HAAR_FIND_BIGGEST_OBJECT*.

7. *CvSize min_size = cvSize* (0,0) yaitu ukuran awal *sub-window* yang digunakan atau ukuran minimal wajah yang dapat dideteksi. Hasil dari *cvHaarDetectObjects* lalu disimpan ke dalam *CvSeq* faces*. Agar hasil dapat terlihat oleh user, maka dapat diberi tanda, biasa berupa persegi ataupun lingkaran, pada setiap wajah yang terdeteksi.

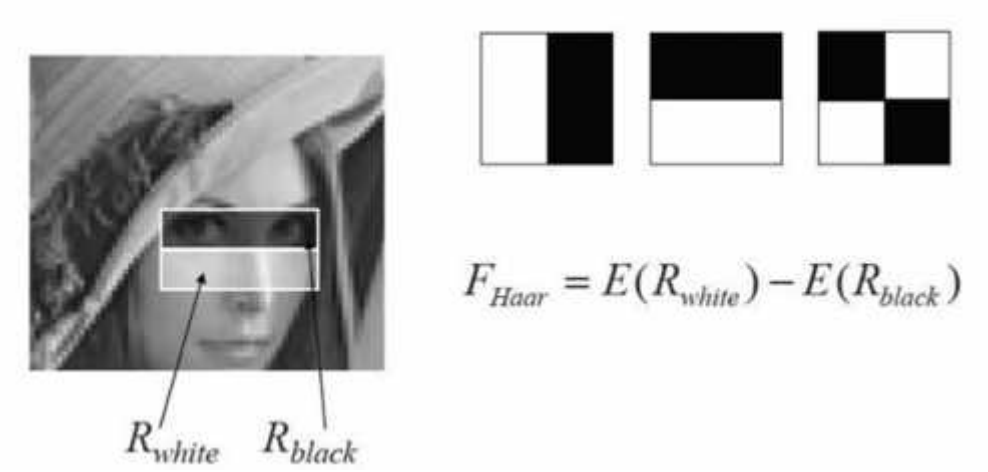
Dalam penelitian ini digunakan sebuah fitur untuk mengenali sebuah objek wajah. Fitur tersebut dinamakan *Haar-Like Feature* yang merupakan suatu fungsi matematika yang berbentuk kotak yang kemudian bersama-sama membentuk *Wavelet family* atau *basis*. *Haar Wavelet* hampir sama dengan fungsi *Fourier* yaitu meletakkan target didalam *interval*, kemudian melakukan proses. *Haar*

Wavelet ini dikenal sebagia *wavelet basis* yang pertama yang dipakai untuk proses pembelajaran yang lainnya. Berikut adalah gambaran bagaimana *Haar Wavelet* bekerja.



Gambar 2.10 Skema Proses Kerja *Haar Wavelets*

Haar-Like feature memproses gambar dalam wilayah kotak-kotak yang berisi beberapa *pixel* dari sebuah bagian gambar. Kemudian *pixel-pixel* dalam satu wilayah tersebut dijumlahkan dan dilakukan proses perhitungan sehingga didapatkan perbedaan dalam setiap wilayah kotak-kotak tersebut. Perbedaan inilah yang dapat dijadikan sebuah kode untuk menandai wilayah tersebut sehingga dapat dilakukan proses bagian gambar yang kita inginkan. Sebagai contoh, misalnya terdapat sebuah *database* yang berisi wajah manusia. Pada umumnya daerah mata akan berwarna lebih gelap sedangkan daerah pipi akan berwarna lebih terang. Dengan menggunakan *Haar-Like feature* dapat dilakukan deteksi daerah mata dan daerah pipi berdasarkan hasil perhitungan perbedaan *pixel* warna gelap dan terang tersebut.



Gambar 2.11 Cara Kerja *Haar Feature*

Gambar diatas menjelaskan proses pendeteksian wajah dengan kondisi gambar diam, namun masalahnya bagaimana proses pendeteksi itu berlangsung bila gambarnya bergerak, dan berjumlah lebih dari satu (jamak), kasus inilah yang mendasari salah satu permasalahan dalam penelitian ini.

Proses dasar yang dilakukan tetap sama yaitu membagi gambar dalam wilayah kotak-kotak dan memprosesnya untuk mendapatkan nilai perbedaan antar wilayah kotak yang satu dengan yang lain, nilai ini biasa disebut dengan *threshold*. Di dalam gambar yang bergerak, proses pencarian *threshold* ini dilakukan terus menerus secara dinamis seiring dengan bergeraknya gambar sehingga pendeteksi tersebut akan terus mendeteksi objek yang ingin di deteksi tersebut, meskipun jumlahnya lebih dari satu. Proses yang berlangsung terus menerus tersebut akan mengindikasikan penjumlahan menjadi semakin banyak dan proses perhitungan akan menjadi semakin lama. Oleh karena itu proses penjumlahan tersebut diganti dengan proses integral sehingga didapatkan hasil dengan lebih cepat.

2.5. Haarcascade *Template File*

Jenis *haarcascade template file* yang penulis gunakan adalah *haarcascade_frontalface_alt.xml*. *Template* ini memiliki 22 tahapan (*stage* 0 sampai *stage* 21). Jumlah tahapan ini menunjukkan jumlah tingkatan *classifier*

bertingkat yang digunakan. Jadi setiap citra *sub-window* akan diseleksi setiap melewati masing-masing tahapan tersebut. Jika *sub-window* tersebut berhasil melewati keseluruhan tahapan, maka akan diasumsikan bahwa *sub-window* tersebut merupakan citra wajah. Untuk lebih jelasnya dapat dilihat potongan *code* dalam *haarcascade_frontalface_alt.xml* sebagai berikut:

```
- <opencv_storage>
- <haarcascade_frontalface_alt type_id="opencv-haar-classifier">
<size>20 20</size>
- <stages>
- <_>
- <!-- stage 0 -->
- <trees>
- <_>
- <!-- tree 0 -->
- <_>
- <!-- root node -->
- <feature>
- <rects>
<_>3 7 14 4 -1.</_>
<_>3 9 14 2 2.</_>
</rects>
<tilted>0</tilted>
</feature>
<threshold>4.0141958743333817e-003</threshold>
<left_val>0.0337941907346249</left_val>
<right_val>0.8378106951713562</right_val>
</_>
</_>
- <_>
- <!-- tree 1 -->
- <_>
- <!-- root node -->
- <feature>
- <rects>
<_>1 2 18 4 -1.</_>
<_>7 2 6 4 3.</_>
</rects>
<tilted>0</tilted>
</feature>
<threshold>0.0151513395830989</threshold>
<left_val>0.1514132022857666</left_val>
<right_val>0.7488812208175659</right_val>
</_>
```

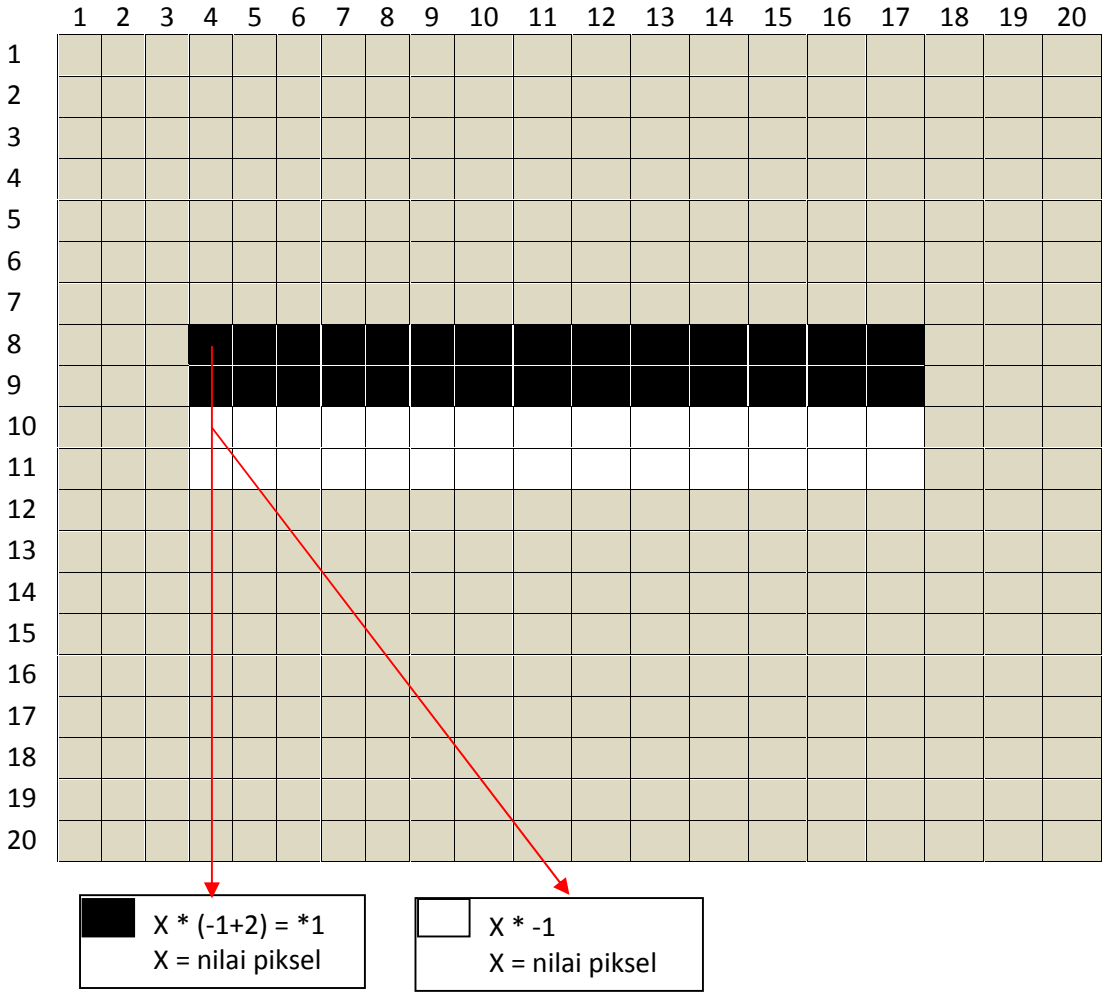
Gambar 2.12 Potongan Kode dalam *haarcascade_frontalface_alt.xml*

```
</_>
- <_>
- <!-- tree 2 -->
- <_>
- <!-- root node -->
- <feature>
- <rects>
<_>1 7 15 9 -1.</_>
<_>1 10 15 3 3.</_>
</rects>
<tilted>0</tilted>
</feature>
<threshold>4.2109931819140911e-003</threshold>
<left_val>0.0900492817163467</left_val>
<right_val>0.6374819874763489</right_val>
</_>
</_>
</trees>
<stage_threshold>0.8226894140243530</stage_threshold>
<parent>-1</parent>
<next>-1</next>
</_>
- <_>
- <!-- stage 1 -->
...
<!-- stage 21 -->
...
</_>
</stages>
</haarcascade_frontalface_alt>
</opencv_storage>
```

Gambar 2.12 Potongan Kode dalam *haarcascade_frontalface_alt.xml* (Lanjutan)

Pada bagian awal *code*, terdapat definisi ukuran *window* yang akan digunakan dalam mengklasifikasikan setiap citra *sub-window* yang ada, yaitu pada bagian <size>. Hal ini menunjukkan bahwa setiap citra *sub-window* yang akan diklasifikasi, berapapun ukurannya, akan diubah terlebih dahulu ke dalam ukuran 20 x 20. Misal, terdapat sub-*window* berukuran 40 x 40, maka sub-*window* tersebut akan dikonversi menjadi *window* berukuran 20x20. Demikian pula, jika terdapat sub-*window* yang lebih kecil dari 20 x 20, misalnya 10 x 10, sub-*window* tersebut akan diubah ke dalam ukuran 20 x 20. Dengan melakukan pengkonversian ini, maka citra sub-*window* berukuran sekecil apapun dapat diklasifikasi.

Untuk lebih jelasnya dapat dilihat gambar 2.10 dan gambar 2.11 tentang simulasi bagaimana sebuah fitur terbentuk dari sub-*region* dengan ukuran 20 x 20 piksel yang digunakan dalam penelitian ini.

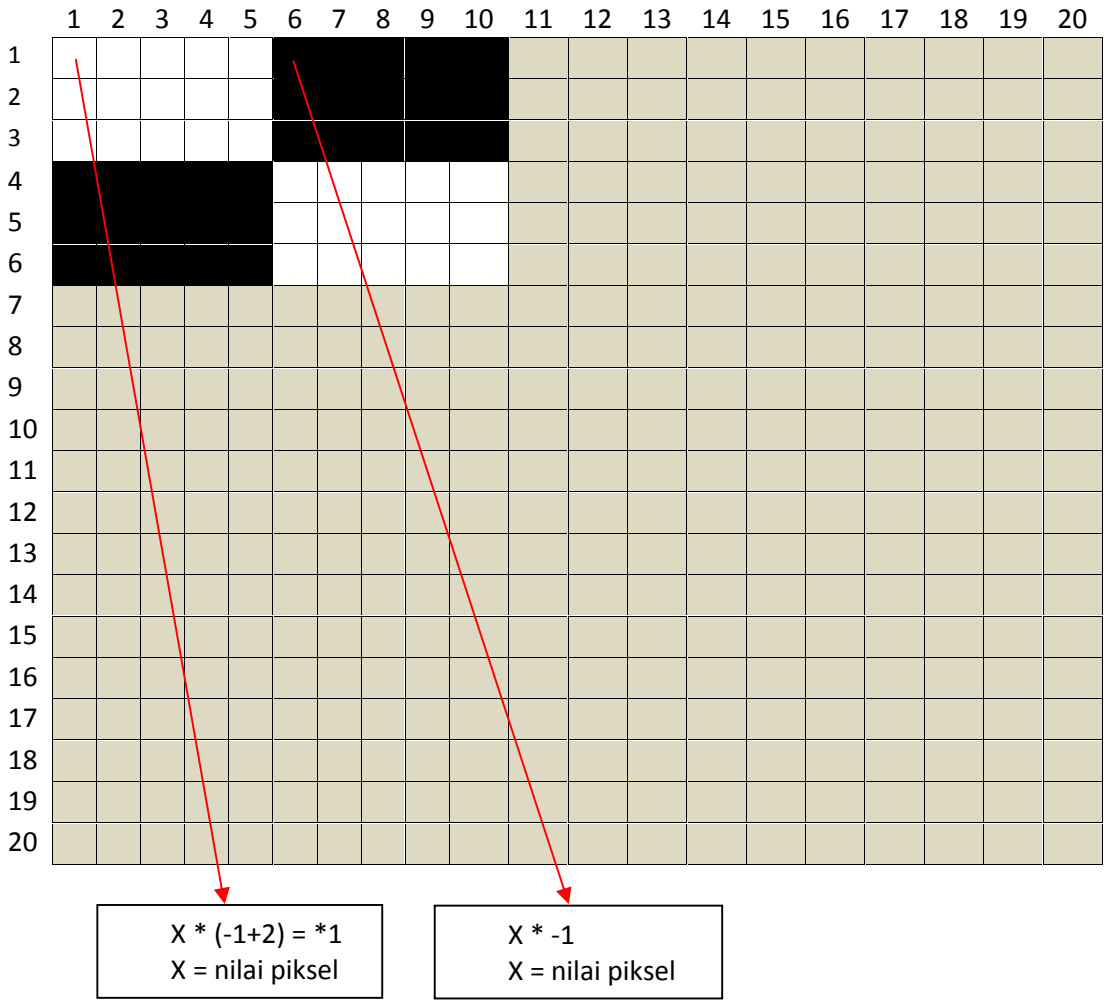


Gambar 2.13 Contoh Penghitungan Nilai Fitur 1

Keterangan: warna hitam dan putih pada piksel menunjukkan posisi fitur yang terdiri dari piksel bernilai positif (putih) dan piksel bernilai negatif (hitam)

Contoh lain:

```
<rects>
<_>0 0 10 6 -1.</_>
<_>0 0 5 3 2.</_>
<_>5 3 5 3 2.</_>
</rects>
```

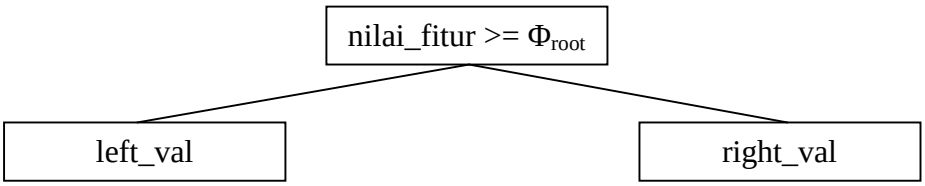


Gambar 2.14 Contoh Penghitungan Nilai Fitur 2

Keterangan: warna hitam dan putih pada piksel menunjukkan posisi fitur yang terdiri dari piksel bernilai positif (putih) dan piksel bernilai negatif (hitam)

Citra *sub-window* kemudian di-*threshold* sebesar nilai yang tertera pada bagian *threshold.* Kemudian, dengan menggunakan nilai piksel dari citra sub-

window yang telah di-*threshold* tersebut, dilakukan perhitungan fitur *Haarlike*. Hasil perhitungan tersebut lalu dibandingkan dengan nilai *threshold*, apabila nilai fitur lebih kecil dari nilai *threshold*, maka fitur tersebut akan dianggap tidak ada. Sebalinya, jika nilai fitur lebih besar atau sama dengan nilai *threshold*, maka nilai fitur ini akan dilanjutkan ke *left_val* dan *right_val*. Apabila nilai fitur berada di antara nilai *left_val* dan *right_val*, maka *sub-window* tersebut berhasil melewati *classifier* lemah atau *tree* tersebut.

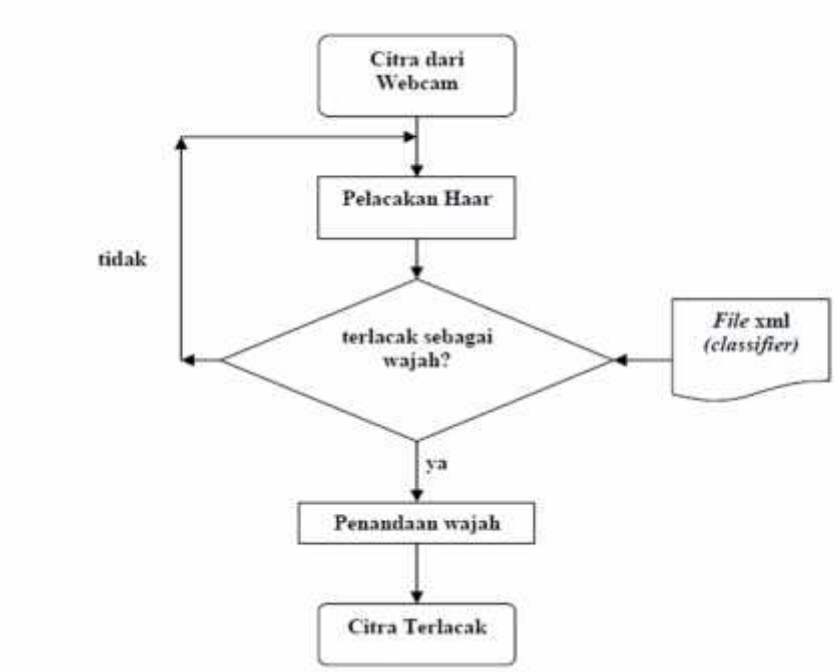


Gambar 2.15 Pohon keputusan dalam *haarcascade_frontalface_alt.xml*

Demikianlah proses pengklasifikasian yang terjadi di dalam setiap *tree*. Apabila suatu *sub-window* berhasil melewati seluruh *tree* di dalam suatu tahapan, maka *sub-window* ini dinyatakan berhasil melewati satu tingkat *classifier*, dan akan dilanjutkan ke tingkat *classifier* selanjutnya. Sebaliknya, jika *sub-window* tersebut gagal melewati suatu *tree*, maka *sub-window* langsung dieliminasi, dan dianggap sebagai bukan wajah.

2.6. Teori Pengujian pada *Face Detection*

Pengujian pelacakan wajah dengan melakukan proses pelacakan dan dengan menerapkan kondisi posisi wajah, tingkat pencahayaan, perbedaan ukuran citra yang dipengaruhi jarak obyek (wajah), faktor keberadaan komponen struktural atau penghalang, juga ekspresi wajah yang berbeda, sehingga akan diketahui sejauh mana pelacakan dapat berhasil dilakukan, dan didapat pula batasan sejauh mana proses pelacakan dapat dilakukan. Gambar 2.8 berikut adalah *flowchart* proses *face detection* dilakukan dengan *classifier file* *.xml:



Gambar 2.16 *Flowchart* Algoritma Proses Pelacakan Wajah dengan *classifier file* *.xml

Keberhasilan dalam mendeteksi wajah akan dipengaruhi oleh posisi wajah dengan *file* xml ini diseting untuk maksimum rotasi sudut terhadap sumbu x adalah 0.6, sumbu y adalah 0,6 dan terhadap sumbu z adalah 0.3 untuk toleransi dalam pelacakan wajah (Seo, 2007). Nilai seting ini sebagai penetapan kemampuan sistem dalam pelacakan wajah untuk toleransi maksimum.

Pada tahap selanjutnya proses ini akan diteruskan pada tahap pengenalan wajah atau *face recognition* yang akan dibahas pada sub bab berikutnya.

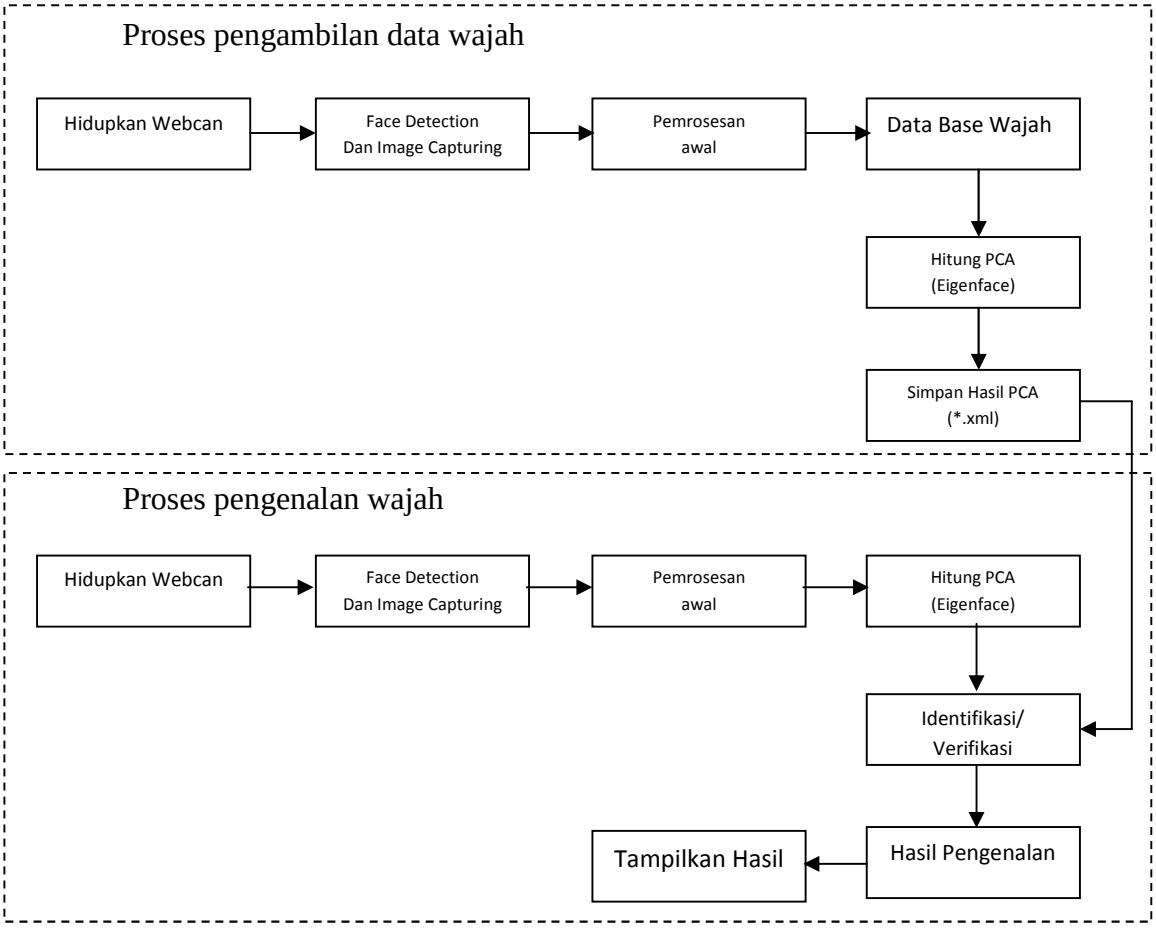
2.7. *Face Recognition*

Wajah merupakan bagian tubuh yang berperanan penting dalam proses penyampaian ciri, identitas dan emosi seseorang. Kemampuan manusia dalam mengenali wajah seseorang sering terjadi secara tidak sadar, manusia mampu mengenali ribuan wajah manusia dan mengidentifikasi wajah yang sekilas dikenalnya sampai beberapa tahun kemudian. Proses ini berlansung begitu cepat dan tersimpan dalam memori manusia dalam waktu cukup lama, walaupun wajah yang dikenalinya dapat terjadi perubahan visual seperti perubahan kondisi,

ekspresi, sudut pandang, penuaan dan penambahan aksesoris seperti kaca mata, topi dan bahkan perubahan model rambut (Turk, M., Pertland, A. 1991:1). Jadi wajah merupakan sebuah patokan yang menjadi indikasi untuk mengenali seseorang.

Face Recognition adalah sebuah proses lanjutan dari proses *Face Detection*. Di dalam *Face Detection* terdapat proses mendeteksi bagian wajah dari seseorang, wajah tersebut bisa didapatkan dari gambar maupun *video*. Banyak metode yang bisa digunakan untuk mendeteksi wajah tersebut, salah satu metode yang paling terkenal adalah dengan memanfaatkan hasil *training* dari *haarcascade*. *Haarcascade* tersebut berupa *file* xml (.xml) dan biasanya terinstall secara *default* ketika kita melakukan installasi *OpenCV*. *File* xml dari *haarcasscade* tersebut bermacam-macam jenisnya. Beberapa contohnya adalah *haarcascade frontalface alt.xml* atau *haarcascade eye.xml*. *Frontalface* merupakan contoh jenis *haarcascade* yang digunakan untuk mendeteksi wajah sedangkan *Eye* merupakan contoh jenis *haarcascade* yang digunakan untuk mendeteksi mata. Melihat kedua contoh tersebut, sebenarnya penggunaan *Face Detection* sangat sederhana apabila telah mengerti prinsip dasar dari *haarcascade* dan cara kerjanya karena sebenarnya yang dilakukan hanyalah menggunakan *file* xml tersebut dan mengaturnya di dalam program yang dibuat sesuai dengan kebutuhan yang diinginkan. Untuk mendeteksi mata, maka caranya secara umum hampir sama karena kita hanya tinggal mengganti *file* xml tersebut (Nathanael, dkk. 2012).

Dalam proses pengenalan wajah secara garis besar terdapat dua tahapan utama, yaitu proses pengambilan data wajah yang akan dikenali sistem dan proses pencocokan data wajah yang tersimpan dengan data wajah yang baru untuk dilakukan pengenalan wajah. Lebih jelasnya dapat dilihat pada bagan diagram blok gambar 2.9 berikut ini:



Gambar 2.17 *Block Diagram System* Pengenalan Wajah

Gambar 2.9 merupakan *block diagram software* untuk melakukan pengenalan dari sistem yang digunakan, dimana jalannya sistem dibagi menjadi 2, yaitu *block* pengambilan data dan *block* pengenalan wajah. Jalannya *sistem block* pengambilan data adalah sebagai berikut :

1. Aktifkan webcam untuk menampilkan gambar yang ditangkap webcam kedalam aplikasi.
2. Penangkapan citra wajah (*image capturing*) dapat dilakukan secara langsung (*real time*) menggunakan *webcam*, setelah terdeteksi adanya gambar wajah pada tampilan *window* dari *webcam*.
3. Kemudian dilakukan proses pemrosesan awal yang meliputi, normalisasi ukuran citra, *RGB* ke *grayscale*, *histogram equalization* untuk perataan warna agar lebih sederhana.
4. Simpan data wajah yang diambil dalam bentuk *.pgm.

5. Kemudian dilakukan proses *PCA* untuk mengutip bagian terpenting dengan metode *eigenface* sehingga didapatkan *eigenvector* dan *eigenvalue* dari gambar tersebut.
6. Proses penyimpanan kedalam data wajah untuk setiap citra wajah yang telah digunakan dalam proses *PCA* dalam bentuk *.xml , semakin kompleks dan sering maka proses pengenalan wajah akan semakin baik.
7. Data yang telah disimpan nantinya digunakan sebagai nilai pembanding pada proses penghitungan jarak untuk pengenalan wajah.

Sedangkan untuk proses pengenalan adalah sebagai berikut :

1. Aktifkan *webcam* untuk menampilkan gambar yang ditangkap webcam kedalam aplikasi.
2. Penangkapan citra wajah (*image capturing*) dapat dilakukan secara langsung (*real time*) menggunakan webcam, setelah terdeteksi adanya gambar wajah pada tampilan *window* dari *webcam*.
3. Kemudian dilakukan proses pemrosesan awal yang meliputi, normalisasi ukuran citra, *RGB* ke *grayscale*, *histogram equalization* untuk perataan warna agar lebih sederhana.
4. Kemudian dilakukan proses *PCA* untuk mengutip bagian terpenting dengan metode *eigenface* sehingga didapatkan e*igenvector* dan *eigenvalue* dari gambar tersebut.
5. Proses pengenalan wajah dengan menghitung jarak antara fitur wajah yang ada dalam data dan fitur wajah yang baru. Jarak yang didapat dicari yang terkecil untuk identifikasi, sehingga kemudian diperoleh hasil pengenalan wajah.

2.8. *Histogram Equalization*

Histogram Equalization adalah suatu proses untuk meratakan histogram agar derajat keabuan dari yang paling rendah (0) sampai dengan yang paling tinggi (255) mempunyai kemunculan yang rata. Dengan *histogram equalization* hasil gambar yang memiliki histogram yang tidak merata atau distribusi kumulatif

yang banyak loncatan gradiasinya akan menjadi gambar yang lebih jelas karena derajat keabuannya tidak dominan gelap atau dominan terang, namun warna citra hitam atau putih yang merepresentasikan notasi biner. Penentuan pewarnaan hitam atau putih ditentukan oleh *threshold* yang menjadi acuan dominasi kedua warna tersebut. Proses *histogram equalization* ini menggunakan distribusi kumulatif, karena dalam proses ini dilakakukan perataan gradient dari distribusi kumulatifnya (Kurniawan, agus, dkk. 2009).

2.9. Metode *Eigenface* Berbasis PCA (*Prinsipal Component Analysis*)

Eigen face adalah salah satu algoritma pengenalan wajah yang berdasarkan pada *Principle Component Analysis* (*PCA*) yang dikembangkan di MIT. Algoritma *Eigenface* secara keseluruhan cukup sederhana. *Training Image* direpresentasikan dalam sebuah vektor flat (gabungan vektor) dan digabung bersama-sama menjadi sebuah matriks tunggal. *Eigen Vector* kemudian diekstraksi dan disimpan dalam *file temporary* atau *database*. *Training image* kemudian kemudian diproyeksikan dalam *feature space,* dinamai *face space* yang ditentukan oleh *eigen vector*.

Eigenface adalah sebuah metode *face recognition* yang mudah untuk diimplementasikan. Biasanya *eigenface* ini digunakan sebagai bahan pembelajaran karena merupakan metode pertama yang digunakan untuk mendeteksi benda sehingga metode ini yang paling sering digunakan. Jika dimisalkan dalam suatu sistem sudah ada *database* yang berisi gambar-gambar dari orang yang dikenali, kemudian sistem diberikan gambar orang yang tidak dikenali maka secara umum berikut adalah langkah atau prosedur sederhana sebuah wajah terdeteksi:

1. Menghitung jarak dari gambar tersebut dibandingkan dengan gambar-gambar yang ada di dalam *database*.
2. Memilih sebuah gambar dari *database* yang mendekati wajah yang ada di dalam gambar tersebut.

3. Jika jarak yang telah diukur tersebut hasilnya diatas nilai dari *threshold* maka gambar tersebut dikenali oleh sistem, tetapi bila nilai yang dihasilkan lebih kecil maka gambar tersebut termasuk dalam gambar yang tidak dikenali oleh sistem karena sistem hanya mengenali gambar yang ada di dalam *database* (Hewit, Robin, 2007).

Jadi proses yang menerapkan metode *eigenface* ini adalah untuk pengambilan informasi ciri dari citra, metode ini selanjutnya diproses berdasarkan algoritma *PCA* (*Principal Component Analysis*) (Alfin Soleh, 2013). Kemudian nilai hasil kalkulasi dari proses tersebut akan diproses pada kalkulasi berikutnya untuk mendapatkan nilai kesamaan jarak (*Similarity Distance Measure*) dengan menggunakan metode *Eucledian Distance* yang akan dibahas pada pokok bahasan berikutnya.

Pada penelitian ini penulis menggunakan algoritma *PCA* karena lebih sederhana dalam proses kalkulasi sehingga dapat memberikan efisiensi waktu terhadap program yang dibuat. PCA digunakan untuk mereduksi dari sekumpulan ruang citra sehingga bisa mendapatkan ruang wajah lebih sederhana.

Prinsip dasar dari algoritma *PCA* adalah memproyeksikan citra ke dalam bidang ruang *eigen*-nya. Caranya adalah dengan mencari *eigen vector* yang dimiliki setiap citra dan memproyeksikannya ke dalam ruang *eigen* yang didapat tersebut. Besarnya dimensi ruang *eigen* tergantung dari jumlah citra yang dimiliki oleh program *training*.

Sebuah citra wajah dapat dilihat sebagai sebuah vektor yang jika panjangnya W dan H piksel, maka jumlah komponen dari vektor ini adalah W x H (setiap piksel dikodekan oleh satu komponen vektor) dan vektor wajah tersebut berada dalam ruang wajah (ruang-*eigen*) yang merupakan ruang dari semua citra.

Namun keseluruhan ruang citra bukanlah ruang yang optimal untuk menggambarkan wajah, oleh karena itu dibutuhkan cara yang bertujuan untuk membentuk sebuah ruang wajah yang dapat menggambarkan wajah dengan lebih baik. PCA digunakan untuk mereduksi dimensi dari sekumpulan atau ruang citra, sehingga bisa mendapatkan ruang wajah yang lebih baik. Berikut ini langkah-langkah pencarian *eigenface* dan fitur *PCA*-nya:

1. Sebagai contoh, terdapat citra 2D berukuran 3x4 piksel dalam data *training*. Mulanya dilakukan normalisasi dengan merubah citra 2D menjadi citra 1D dan disimpan kedalam matriks *lm* = [Γ_1, Γ_2,… Γ_m]. Dimana *lm* merupakan matriks baru yang berisi nilai dari seluruh data, Γ_i merupakan data ke-i.
2. Kemudian cari nilai rata-rata (ψ) dari tiap baris yang ada dalam matriks *lm*, berikut persamaan yang dipakai:

   $\psi = \frac{1}{M} \sum_{n=1}^{M} \Gamma_n$ 2.8

   Dimana ψ merupakan nilai rata-rata, M adalah banyak data yang direferensi, dan Γ_n merupakan nilai data ke-n.
3. Setelah itu kurangi tiap nilai yang ada dalam matriks *lm* dengan nilai rata-rata yang sesuai dengan baris hasil pengurangannya dan hasil dari pengurangan tersebut dimasukan ke dalam matriks Φ untuk mendapatkan fitur PCA atau ciri data, berikut adalah persamaannya;

   $\Phi_i = \Gamma_i - \psi$ ……...…...……..2.9

   Dimana Φ_i adalah nilai pola hasil ekstraksi data ke-i, Γ_I adalah nilai data ke-i, dan ψ adalah nilai *mean* atau nilai rata-rata.
4. Setelah mendapatkan matriks Φ, lalu hitung matriks kovarian C dengan cara mengalikan matriks Φ dengan transposenya (A^T). Berikut persamaannya:

   $C = A \times A^T$…….2.10

   Dimana C merupakan nilai *covariance matrix*, A adalah { $\Phi_1, \Phi_2, \ldots \Phi_M$ }, dan A^T adalah nialai *transpose* dari dari A.
5. Selanjutnya yaitu menghitung nilai *eigen* (λ) dari matriks kovarian C. Dengan persamaan:

   $Det[\lambda - C] = 0$……..2.11
6. Langkah selanjutnya adalah menghitung vektor *eigen* emnggunakan persamaan :

   $AX = \lambda X$……..2.12

   Setelah itu menghilangkan nilai kolom vektor *eigen* yang nilainya lebih

kecil dari ambang batas, kemudian adalah menghitung nilai *eigenfaces* (Ef) dengan persamaan:

Ef = fitur^{T} x vector eigen…….……....2.13

7. Setelah nilai *eigenface* diperoleh, proses selanjutnya adalah menghitung nilai bobot masing-masing citra referensi dengan persamaan:

   W = fitur x EfN…….2.14

   Dimana W adalah nilai bobot citra referensi dan EfN adalah nilai *eigenface* terbesar.
8. Langkah selanjutnya adalah melakukan proses ekstraksi ciri citra uji. Proses ini dilakukan sama persis pada proses sebelumnya hingga ditemukan nilai bobot dari citra uji tersebut yang dilambangkan dengan Wx, dengan persamaan:

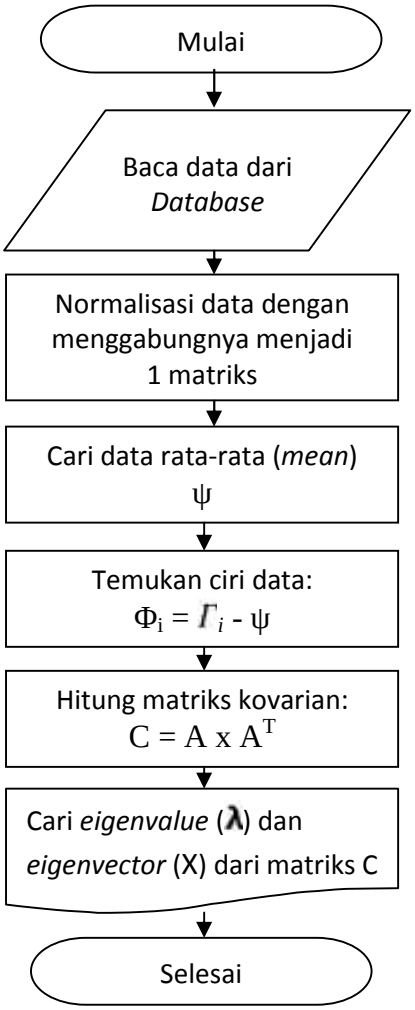
   Wx = (fitur x)(EfN)…….2.15
9. Langkah terakhir yaitu proses pengenalan wajah menggunakan perhitungan *eucledien distance* dengan persamaan:

   D (x,y) = $\overline{|x - y|^2}$ …….……...……2.16

   Dimana D adalah nilai *similarity distance* atau nilai jarak kemiripan, x adalah nilai fitur citra referensi dan y adalah nilai fitur citra uji.
10. Nilai perhitungan yang diambil sebagai hasil dari pengenalan wajah adalah yang terkecil (Alfin Soleh, 2013).

Eigenvalue memiliki fungsi untuk memberikan seberapa besar deformasi bidang diberikan, sedangkan *eigenvector* berfungsi memberikan informasi arah perubahan deformasi tersebut. Informasi kedua nilai tersebut yang berisi informasi citra wajah yang disebut *eigenface.*

Serangkaian proses dalam menentukan *eigenvalue* dan *eigenvector* dapat dilihat pada gambar alur 2.10 berikut ini:



Gambar 2.18 Menentukan *Eigenvalue* dan *Eigenvector*

2.10. *Distance Measures*

Proses ini merupakan proses akhir pengenalan wajah, inti dari proses ini adalah pencocokan dua citra wajah antara citra data *training* dan citra yang diuji. Pencocokan tersebut salah satunya dengan menggunakan metode pengukuran jarak. Pada dasarnya pengukuran jarak digunakan untuk menghitung perbedaan antara dua vektor citra dalam ruang *eigen*. Setelah citra wajah diproyeksikan ke dalam *space* wajah, tugas selanjutnya adalah menentukan citra wajah yang mana yang paling mirip dengan citra dalam data *training*. *Distance-Based Similarity*

Measure digunakan untuk mengukur tingkat kesamaan dua buah objek dari segi jarak geometris dari variabel-variabel yang tercakup di dalam kedua objek tersebut. Metode penghitungan ini memiliki kelebihan tertentu, misalnya jarak antara dua benda tidak dipengaruhi oleh penambahan objek baru untuk analisis. Namun, jarak dapat sangat dipengaruhi oleh perbedaan skala antara dimensi dari mana jarak dihitung. Dalam penggunaannya terdapat banyak metode dalam mengukur tingkat kesamaan jarak, diantaranya yaitu jarak *euclidean, manhattan (city block), mahalanobis, chebyshev, correlation, angle-based, minkowski dan cosini-correlation* (Alfin Soleh, 2013).

Dalam penggunaannya dengan algoritma *eigenface*,perhitungan *eucledian distance* digunakan untuk menghitung jarak antara citra data *training* yang diproyeksikan dan citra yang diuji, yang juga diproyeksikan dengan menggunakan persamaan masing-masing. Pada akhirnya, masing-masing perhitungan memiliki nilai dengan kelebihan dan kekurangannya masing-masing.

2.11. *Eucledian Distance*

Secara umum dalam hal perhitunagn jarak yang menggunakan *matrix*, formulasi yang digunakan adalah berdasarkan algoritma *Minkowski Distance of Order*. Dimana dalam kalkulasinya tingkat keakuratan hasil perhitungan ditentukan oleh nilai akar ciri atau *eigenvalue*, secara sederhana *eigenvalue* merupakan suatu nilai yang menunjukkan seberapa besar pengaruh suatu variable terhadap pembentukan karakteristik sebuah *vector* atau *matrix*, *eigenvalue* ini dinotasikan dengan λ (*Lambda*).

Formulanya adalah sebagai berikut :

$$d_{ij} = \sqrt[\lambda]{\sum_{k=1}^{n} |x_{ik} - x_{jk}|^{\lambda}} \quad \text{.....2.17}$$

Dimana variabel k berfungsi menunjukkan nilai koordinat yang merepresentasikan nilai *feature,* sedangkan x_i dan x_j merupakan nilai *pair object* yang akan dihitung nilai jarak perbandingan diantara keduanya. Kemudian, jika algoritma ini dekembangkan lagi dengan memberikan nilai λ = 1, maka disebut dengan algoritma *Manhattan distance*, sedangkan jika nilai λ =

2, maka disebut dengan algoritma *Eucledien distance.* Secara proses dalam kalkulasi algoritma *Manhattan distance* lebih sederhana karena kalkulasinya tanpa ada proses pencarian akar dan pangkat, karena keduanya bernilai 1. Sedangkan algoritma *Minkowski Distance of Order* prosesnya terlalu rumit karena nlai λ dapat bernilai berapa saja (Teknomo, kardi, 2006).

Dalam implementasi algoritma *Eucledien distance* dan algoritma *Minkowski Distance of Order* sering digunakan dalam pencocokan posisi suatu objek. Setiap algoritma diterapkan kedalam penyelesaian kasus sesuai kebutuhan penelitian tersebut. Pada kasus pengenalan wajah dalam peneitian ini penulis memilih menggunakan algoritma *Eucledian distance,* karena secara umum lebih sederhana. Meskipun perhitungan dengan *Manhattan distance* lebih sederhana karena tidak ada proses pembulatan, namun dalam implementasinya metode ini hanya dibutuhkan dalam mengukur jarak antara dua objek dengan mengabaikan posisi, sehingga hasil kurang akurat, sedangkan dengan *eucledian distance* posisi jarak antara kedua objek juga diperhitungkan sehingga nilai sangat akurat akurat dalam mencari kesamaan jarak.

Proses penghitungan jarak ini bertujuan untuk mendapatkan perbedaan jumlah *pixel* dari setiap citra pada gambar wajah, misalnya dalam dua gambar wajah yang berbeda terdapat 2.500 perbedaan *pixel*. Kita ingin menjadikan 2.500 nilai tersebut menjadi satu nilai untuk kemudian kita lakukan proses lebih lanjut. (Hewit, Robin, 2007).

Untuk formulanya sendiri hasil dalam perhitungan menggunakan *Euclidean Distance* ini hampir menyerupai dengan hasil perhitungan *Manhattan distance*. Pengukuran ini lebih dikenal dengan *city block distance*. Perbedaannya pada pangkat dan akar. Berikut ini adalah formulanya:

D (x,y) = $\sqrt{|x - y|^2}$ ……..……..2.18

Keterangan:

D (x,y) = nilai kesamaan jarak antara citra uji x dengan citra data *training* y

x = nilai ciri citra yang diuji

y = nilai ciri citra data *training*

Nilai x adalah nilai ciri dari citra yang diuji yang di-*capture* dari

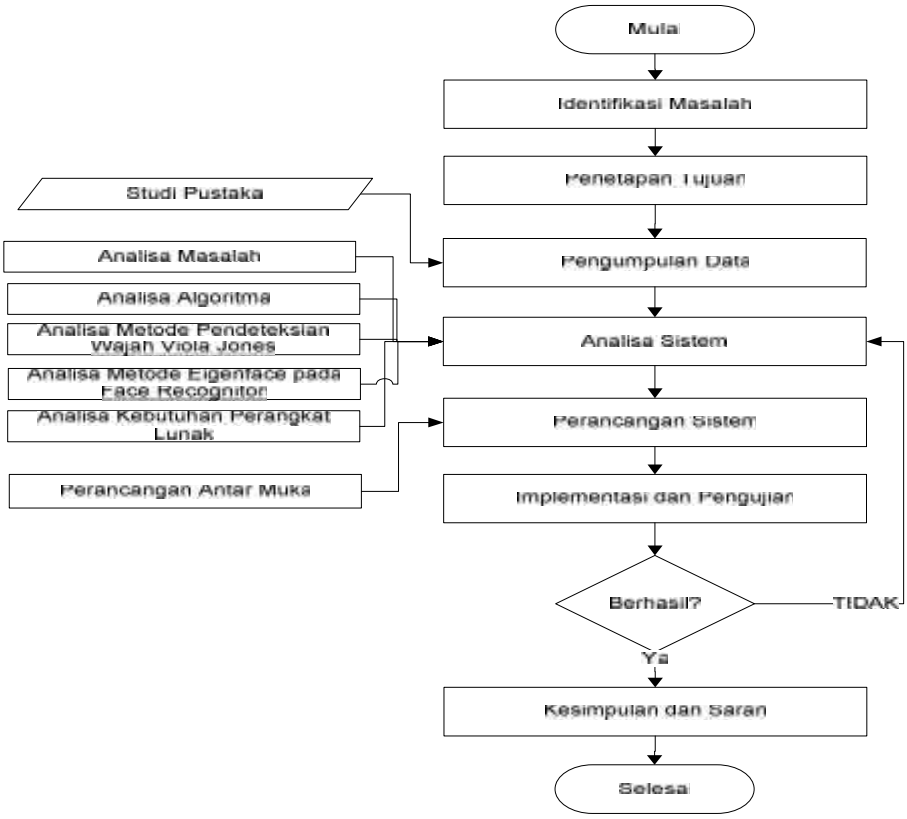
webcam secara *real time*, sedangkan nilai y pada penelitian ini adalah nilai ciri dari data *training* yang sudah ada sebelumnya didalam *database*. Perhitungan *eucledian distance* memiliki kelebihan dibandingkan perhitungan kesamaan jarak lainnya yaitu perhitungan ini lebih akurat namun sederhana. (Teknomo, kardi, 2006). Pada penelitian ini, hasil perhitungan *eucledian distance* yang menjadi hasil sistem pengenalan wajah adalah nilai yang terkecil. Sedangkan hasil akhir dari penelitian mengenai sistem pengenalan wajah ini yaitu berupa citra wajah.

BAB III
METODOLOGI PENELITIAN

Metodologi adalah tatacara yang disusun secara pasti, sistematis dan logis sebgai landasan untuk suatu kegiatan tertentu. Metodologi yang diperlukan untuk tugas akhir ini terdiri dari beberapa tahap seperti: tahap pengumpulan data, tahap analisa dan perancangan, tahap implementasi dan tahap pengujian.

3.1. Alur Metodologi Penelitian

Dalam sebuah penelitian dibutuhkan gambaran alur dari proses-proses yang diperlukan mulai dari tahap awal hingga hasil, agar tahapan dalam melakukan penelitian tersebut berjalan dengan terstruktur, sehingga dapat dilakukan kontrol dengan mudah jika terjadi suatu kendala. Dalam penelitian ini penulis melakukan tahapan metodologi penelitian seperti gambar alur dibawah ini:



Gambar 3.1 Alur Metodologi Penelitian

Berdasarkan gambar 3.1 diatas, dapat diketahui terdapat sekurang-kurangnya terdapat tujuh tahapan umum yang penulis lakukan untuk melakukan serangkaian penelitian dalam tugas akhir ini, yaitu meliputi: identifikasi masalah, penetapan tujuan, pengumpulan data, analisa sistem, perancangan sistem, implementasi dan pengujian serta kesimpulan dan saran.

3.2. Identifikasi Masalah

Identifikasi masalah merupakan salah satu tahapan penelitian dalam menentukan serta mengidentifikasi masalah-masalah yang berhubungan dengan penelitian, rincian identifikasi masalah dalam penelitian ini adalah sebagai berikut:

1. Menganalisa serta mengimplementasikan metode *Viola Jones* dalam *multi-face detection.*
2. Menganalisa serta mengimplementasikan metode *eigenface* berbasis *principle component analysis* (*PCA*) dalam proses pengenalan wajah (*face recognition*).
3. Menguji performa serta keakuratan yang dihasilkan oleh sistem dalam menggunakan kedua metode diatas pada proses pengenalan wajah manusia (*face recognition*).
4. Menghasilkan kesimpulan dan rekomendasi terhadap hasil setelah dilakukan proses implementasi dan pengujian aplikasi.

3.3. Penetapan Tujuan

Penetapan tujuan merupakan sebuah proses untuk menentukan fokus utama yang dilakukan dalam penelitian yang menjadi acuan atau patokan agar tidak terjadi kesalahan atau penyimpangan terhadap hasil penelitian. Berdasarkan pada identifikasi masalah diatas, maka dapat ditarik secara garis besar yang menjadi tujuan utama dalam penelitian ini adalah menganalisa, mengimplemantasikan serta merancang dan membangun sebuah aplikasi pendeteksian wajah dengan kemampuan secara plural menggunakan metode *Viola*

Jones, serta mampu melakukan identifikasi seteiap wajah yang dideteksi menggunakan metode *eigenface* berbasis *PCA.*

3.4. Pengumpulan Data

Pengumpulan data merupakan salah satu rangkaian proses dalam penelitian ini yang bertujuan untuk memperoleh data-data serta informasi-informasi terhadap kasus yang menjadi permasalahan dalam tugas akhir ini. Informasi yang sangat penulis butuhkan dalam penelitian ini adalah informasi-informasi mengenai metode *Viola Jones* dalam sebagai dasar dalam *face detector* dan algoritma *Principal Component Analysis (PCA)* sebagai dasar dalam pengenalan wajah. Dalam memperoleh informasi atau pengumpulan data tersebut penulis melakukan dua pendekatan, diantaranya adalah:

a. Studi Pustaka

   Melakukan studi kepustakaan terhadap berbagai referensi yang berkaitan dengan penelitian yang dilakukan. Topik-topik yang akan dikaji antara lain meliputi: pengenalan pola, pengolahan citra digital, pendeteksian objek secara umum, pendeteksian wajah, dan jaringan syaraf tiruan.

b. Diskusi

   Pendekatan ini dilakukan untuk berdiskusi secara lansung dengan orang-orang yang memahami baik sebagian maupun keseluruhan dari kasus yang dibahas dalam laporan ini tentang bagaimana masalah serta solusi untuk perancangan dan analisa dari aplikasi yang diteliti ini.

3.5. Analisa dan Perancangan Sistem

Analisa dan perancangan merupakan metode yang dilakukan setelah pengumpulan terhadap data-data atau informasi mengenai kasus yang diangkat pada penelitian tugas akhir ini. Analisa berarti metode yang khusus untuk menganalisis masalah yang dapat dimulai dari analisa terhadap alur-alur proses perancangan aplikasi untuk implementasi *multi-face detection*, kemudian menganalisa model hingga rancang bangun aplikasi itu sendiri. Sementara

perancangan berarti metode yang khusus digunakan untuk merancang hal-hal yang telah dianalisa dengan tujuan untuk memberikan kemudahan dan menyederhanakan suatu proses atau jalannya aliran data, perancangan terhadap model, dan merancang rancang bangun aplikasi ini. Perancangan ini meliputi perancangan model sistem yang teridiri dari: *Use Case Diagram, Use Case Scenario, Activity Diagram, Sequence Diagram,* dan *Collaboration Diagram,* serta perancangan *interface* sistem yang terdiri dari *Prototype* sistem.

3.6. Implementasi dan Pengujian

Implementasi dan pengujian merupakan metode terakhir yang digunakan setelah analisa dan perancangan rancang bangun aplikasi selesai dilakukan. Metode ini akan menjelaskan tentang penerapan jalannya rancang bangun yang telah dianalisa dan dirancang. Aplikasi yang telah dirancang dan dianalisa selanjutnya diimplementasikan dan dilakukan pengujian untuk mengetahui tingkat keberhasilan aplikasi yang telah dicapai. Implementasi pengembangan aplikasi ini akan dikembangkan pada spesifikasi *hardware* dan *software* berikut:

1. Perangkat keras

   Processor : *AMD Dual-Core Processor C-50 (1.0 GHz)*
   Memori (RAM) : 2.00 GB
   Storage : 320 GB HDD
   VGA : AMD Radeon HD 6250 *Graphics*
   Kamera (*webcam*) : *External Webcam Lexmark X422, resolution up to* 5 *megapixel*

2. Perangkat Lunak

   Sistem operasi : *Windows 7 Ultimate 32-bit Operating System*
   Bahasa pemrograman : *C#* (*C Sharp*) dan xml
   Editor/compiler : *Microsoft Visual Studio 2010 Express*

2. Perancangan : Notepad++ dan *Microsoft Visual C# 2010 Express*

Kemudian untuk tahap melakukan pengujian aplikasi yang telah dibangun meliputi:

1. Pengujian terhadap citra dengan melakukan pendeteksian atau pengenalan wajah berdasarkan parameter tertentu.
2. Pengujian *blackbox* untuk pengujian tingkah laku aplikasi yang telah dirancang.
3. Pengujian terhadap proses *compile* kode program yang telah dirancang pada setiap elemen atau *interface (form).*
4. Pengujian terhadap *error* jika terjadi kesalahan dalam penulisan kode program dengan proses *debugging* yang dapat dilakukan pada *tool* pemrograman *Microsoft Visual C# 2010 Express.*

3.7. Kesimpulan dan Saran

Tahapan kesimpulan dan saran merupakan akhir dari penelitian tugas akhir ini. Tahapan ini berisi tentang kesimpulan dari hasil-hasil penelitian dan pengujian yang telah dilakukan pada penelitian tugas akhir ini, yaitu merancang aplikasi yang mengimplementasikan *multi-face detection* atau pendeteksian wajah manusia dengan berbasis pada metode *Viola Jones*, dan metode *eigenface* berdasasarkan pada algoritma *PCA* (*Principal Component Analysis*) untuk proses lanjutan dalam pengenalan wajah (*face recognition*), juga berisi saran-saran yang dapat mendukung untuk pengembangan aplikasi ini selanjutnya.

BAB IV
ANALISA DAN PERANCANGAN

Analisa memgang peranan yang penting dalam membuat rincian perangkat lunak pada komputer. Analisa merupakan langkah pemahaman persoalan sebelum mengambil tindakan atau keputusan penyelesaian hasil utama, sedangkan tahap perancangan sistem adalah membuat rincian hasil dari analisa menjadi bentuk perancangan agar dapat dipahami dalam menjelelaskan analisanya pada dunia nyata sehingga mendapatkan gambaran tentang analisa yang mudah dimengerti.

4.1. Analisa Sistem

Analisa sistem merupakan penguraian dari suatu sistem yang utuh ke dalam beberapa komponen dengan maksud untuk mengidentifikasikan dan mengevaluasi permasalahan-permasalahan yang terdapat dalam suatu sistem. Dalam membangun perangkat lunak ini dilakukan beberapa tahap analisa yaitu :

a. Menganalisa masalah yang akan dibangun untuk perangkat lunak dengan menganalisa algoritma yang digunakan.
b. Mengumpulkan data yang diperlukan untuk membangun perangkat lunak melalui studi literatur dan observasi serta tutorial yang mendukung.
c. Analisis Kebutuhan Non Fungsional yang merupakan batasan-batasan dari layanan-layanan dan fungsi-fungsi dari sebuah perangkat lunak.
d. Analisis Kebutuhan Fungsional berupa fungsionalitas dan layanan yang terdapat dalam perangkat lunak.

4.1.1 Analisa Masalah

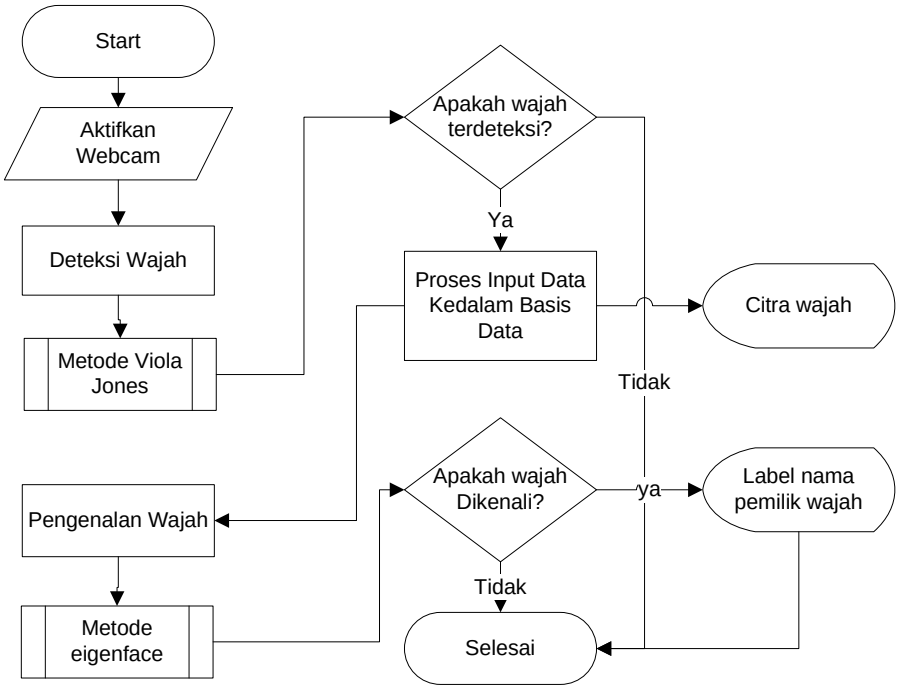
Proses pendeteksian dan pengenalan wajah harus memiliki tingkat akurasi dan ketelitian yang tinggi sehingga dibutuhkan algoritma kompleks yang memiliki kemampuan proses yang baik. Terdapat beberapa masalah yang terdapat dalam media pendeteksian dan pengenalan wajah dalam penelitian ini yaitu :

1. Bagaimana membuktikan tingkat akurasi dan ketelitian metode *Viola Jones* dalam aplikasi pendeteksian wajah dengan kemampuan secara *plural* dengan berberapa kondisi citra wajah berbeda yang dapat mempengaruhi akurasi tersebut.
2. Bagaimana mengimplementasikan serta menguji hasil dari proses pendeteksian wajah dengan menggunakan metode *Viola Jones* tersebut pada proses pengenalan wajah dengan kemampuan secara *plural* menggunakan metode *eigenface*.

4.1.2 Analisa Algoritma

Penelitian ini menggunakan dua metode yang mengkombinasikan dua proses utama, yaitu proses pendeteksian dan pengenalan wajah dengan kemampuan secara *plural*. Metode yang digunakan dalam pendeteksian wajah adalah menggunakan metode *Viola Jones*, sedangkan proses pengenalan wajah menggunakan metode *eigenface* berbasiskan algoritma PCA (*Principal Component Analysis*). Dalam penelitian ini tahap analisa lebih dititikberatkan dalam proses pendeteksian wajah menggunakan metode *Viola Jones*, karena pendeteksian wajah ini merupakan proses utama yang dapat dikembangkan kedalam berbagai penelitian mengenai pemrosesan citra wajah, dalam penelitian ini penulis melakukan pengembangan terhadap proses pengenalan wajah secara *real time*.

Kemudian dari hasil pendeteksian tersebut dilanjutkan ketahap pengenalan wajah dengan mengimplementasikan metode *eigenface*. Bagan proses mulai dari pendeteksian dan pengenalan wajah secara utuh dapat dilihat dalam Gambar 4.1 berikut ini.

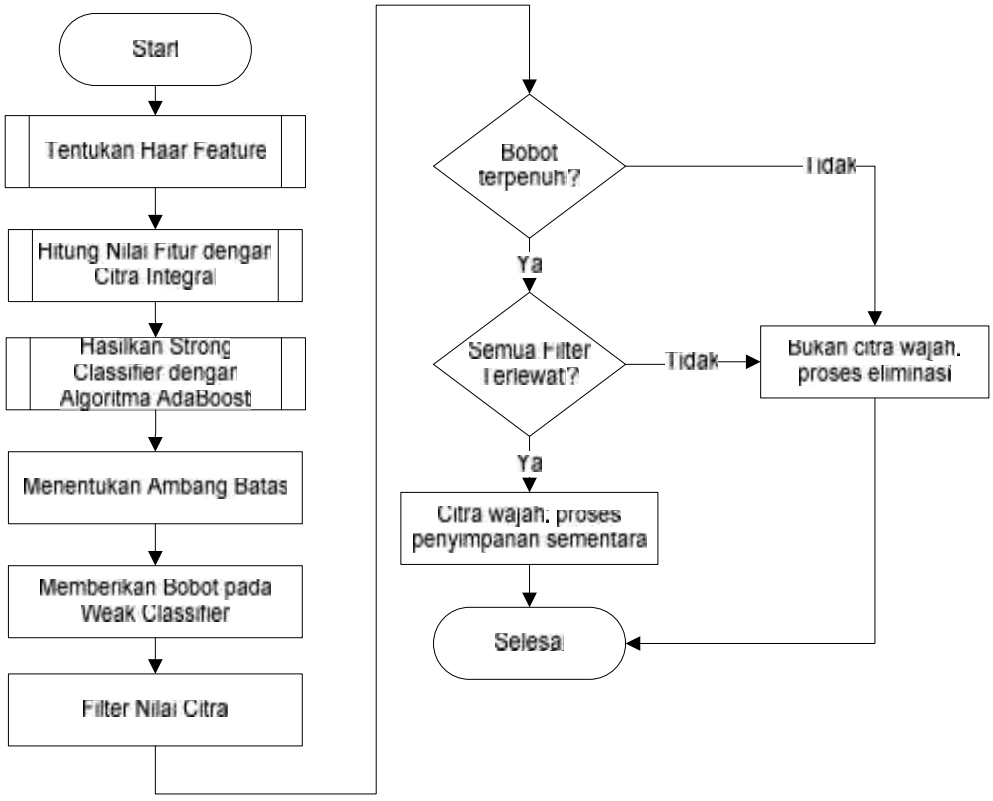


Gambar 4.1 Gambaran Umum Sistem Pengenalan Wajah

Berdasarkan gambar 4.1 diatas dapat diketahui bahwa dalam serangkaian tahapan terdapat beberapa proses utama yang harus dipenuhi dalam sistem pengenalan wajah manusia, yaitu: pemrosesan awal yaitu dimulai dari mengaktifkan *webcam*, kemudian pendeteksian wajah menggunakan metode *Viola Jones*, kemudian penginputan data jika citra wajah terdeteksi, setelah itu proses ekstraksi fitur wajah untuk memperoleh nilai *eigen* menggunakan metode *eigenface*, terakhir yaitu proses pengenalan wajah dengan melakukan perbandingan antara citra *input* dengan citra yang telah di-*training* (*trained faces*) menggunakan metode *Euclidean Distance.* Kemudian akan ditampilkan hasil akhir dari aplikasi apakah wajah dikenali atau tidak dengan pelabelan nama pemilik wajah.

4.1.3 Analisa Metode Pendeteksian Wajah *Viola Jones*

Didalam metode *Viola Jones* ini sendiri mempunyai tahapan-tahapan dalam proses pendeteksian wajah. Disetiap tahapan memiliki proses dan algoritma tertentu yang saling berhubungan, seperti yang dapat dilihat pada Gambar 4.2 berikut ini.



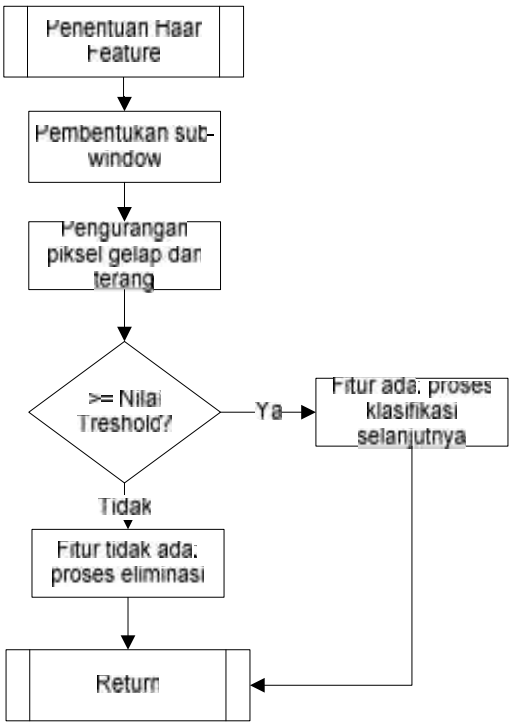
Gambar 4.2 *Flowchart* Pendeteksian Wajah Vioala Jones

Berdasrkan gambar 4.2 diatas dapat dilihat bahawa terdapat 4 proses utama dalam pendeteksian wajah menggunakan metode *Viola Jones*, yaitu: menentukan *Haar Feature*, menghitung nilai fitur dengan citra integral, analisa algoritma *AdaBoost* untuk menentukan apakah fitur berisi wajah dengan serangkaian filter *classifier* dan analisa *Cascaded Classifier* untuk melakukan filter nilai citra dengan memberikan nilai bobot untuk melewati setiap filter sehingga dapat diketahui citra mengandung wajah atau tidak. Berikut ini merupakan penjelasan dari tahapan-tahapan yang ada dalam proses pendeteksian wajah pada metode *Viola Jones* tersebut:

4.1.3.1 Menentukan Fitur pada Proses Pendeteksian Wajah

Metode *Viola Jones* menggunakan data latih dari citra-citra yang kurang tajam sebagai bagian dari proses pengklasifikasian citra. Klasifikasi citra

dilakukan berdasarkan nilai dari sebuah fitur. Untuk dapat memahami bagaimana penentuan nilai fitur ini dapat dilihat pada gambar *flowchart* berikut ini:



Gambar 4.3 *Flowchart* Penentuan Nilai Fitur

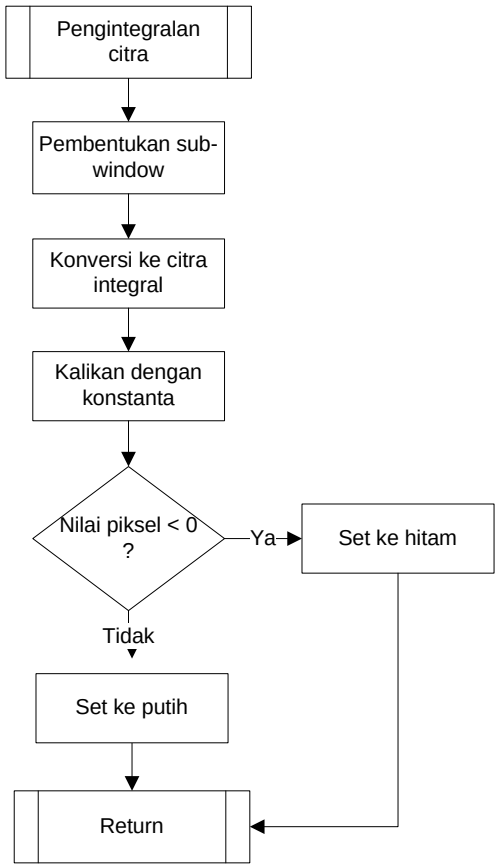
Berdasarkan gambar 4.3 diatas menerangkan bahwa keberadaan ada atau tidaknya fitur wajah ditentukan dengan mengurangi nilai *pixel* di wilayah gelap dengan nilai *pixel* di wilayah terang. Jadi Setiap gambar dirubah kedalam warna hitam dan putih. Jika nilai dari hasil perbedaanya di atas dari ambang batas selama masa pembelajaran citra maka fitur tersebut dapat dikatakan ada.

4.1.3.2 Pemrosesan Citra Integral pada Proses Pendeteksian Wajah

Untuk memudahkan proses perhitungan nilai dari setiap fitur *Haar* pada setiap lokasi gambar dalam penelitian ini dilakukan proses pengintegralan citra. Secara umum integral mempunyai makna menambahkan bobot, bobot merupakan nilai-nilai piksel yang akan ditambahkan ke dalam gambar asli.

Nilai integral dari setiap piksel merupakan jumlah dari semua piksel sebelah atasnya dan di sebelah kirinya. Keseluruhan gambar dapat diintegrasikan dengan operasi bilangan bulat per piksel.

Simulasi perhitungan menggunakan citra integral ini untuk mengetahui efesiensi dan alasan diperlukannya melakukan pengintegralan citra akan dibahas secara terperinci sesuai dengan tahapan yang terdapat dalam gambar *flowchart* 4.4 dibawah ini.



Gambar 4.4 *Flowchart* Pengintegralan Citra

Sebagai simulasi sederhana dimisalkan terdapat citra input berukuran 5x5 dengan nilai masing-masing piksel sebagai berikut:

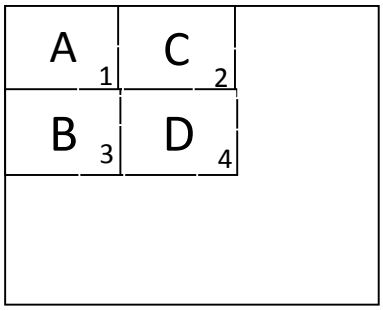
3	6	9	8	7
9	2	7	1	9
5	1	8	7	4
2	4	2	5	9
7	6	2	1	4

Gambar 4.5 Contoh Citra Ukuran 5 x 5 Piksel

Berdasarkan gambar 4.5, maka citra integral dari citra input di atas adalah sebagai berikut:

3	9	18	26	33
12	20	36	45	61
17	26	50	66	86
19	32	58	79	108
26	45	73	95	128

Untuk mengetahui nilai piksel pada setiap elemen matriks dapat dijelaskan berdasarkan pada gambar 4.6 dibawah ini. Nilai piksel D pada citra integral tersebut adalah D = 4 + 1 – (2 + 3). Dimana angka 1-4 merupakan indeks dari nilai masing-masing piksel pada citra dan A-D menunjukkan posisi piksel.



Gambar 4.6 Contoh Pengambilan Piksel

Contoh 1 (berdasarkan hasil citra integral pada gambar 4.5):

A	C
B	D

Citra integral :

3	9
12	2

Nilai piksel pada daerah hitam = D+A-(B+C) = 20+3-(12+9) = 2

Bukti (Lihat citra asli pada gambar 4.5):

3	6
9	2

Nilai piksel pada daerah hitam = 2

Contoh 2 (berdasarkan hasil citra integral pada gambar 4.5):

A	D	G
B	E	H
C	F	I

Citra integral:

3	9	18
12	20	36
17	26	50

Jumlah nilai piksel pada daerah hitam = F+A-(C+D) = 26+3-(17+9) = 3

Bukti (Lihat citra asli pada gambar 4.5):

3	6	9
9	2	7
5	1	8

Jumlah nilai piksel pada daerah hitam = 2+1 = 3

Perbandingan perhitungan nilai fitur *Haarlike* dengan citra integral dan tanpa citra integral, sebagai berikut:

Contoh 1 (berdasarkan hasil citra asli pada gambar 4.5):

Tanpa citra integral:

3	6	9	8	7
9	2	7	1	9
5	1	8	7	4
2	4	2	5	9
7	6	2	1	4

Nilai fitur = | (total piksel hitam) – (total piksel putih) |

= | (3+6+9+2) – (9+8+7+1) |

= | 20 – 25 |

= 5

Dengan citra integral:

3	9	18	26	33
12	20	36	45	61
17	26	50	66	86
19	32	58	79	108
26	45	73	95	128

Nilai fitur = | (total piksel hitam) – (total piksel putih) |

= | [20+0-(0+0)] – [45+0-(20+0)] |

= | 20 – 25 |

= 5

Contoh 2:

Tanpa citra integral:

3	6	9	8	7
9	2	7	1	9
5	1	8	7	4
2	4	2	5	9
7	6	2	1	4

Nilai fitur = | (total piksel hitam) – (total piksel putih) |

= | (2+7+1+8+4+2+6+2) – (1+9+7+4+5+9+1+4) |

= | 32 – 40 |

= 8

Dengan citra integral:

3	9	18	26	33
12	20	36	45	61
17	26	50	66	86
19	32	58	79	108
26	45	73	95	128

Nilai fitur = | (total piksel hitam) – (total piksel putih) |

= | [73+3-(26+18)] – [128+18-(73+33)] |

= | 32 – 40 |

= 8

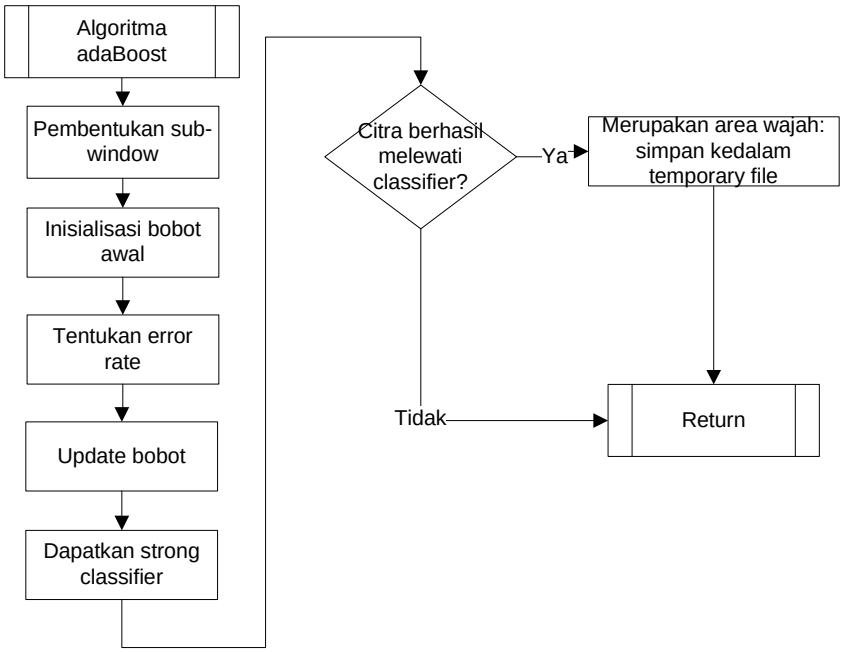
Dari contoh-contoh perbandingan perhitungan fitur di atas, dapat dilihat adanya perbedaan efisiensi perhitungan dengan dan tanpa citra integral. Tanpa menggunakan citra integral, harus dilakukan perhitungan terhadap seluruh nilai piksel yang terdapat di dalam fitur, sedangkan dengan menggunakan citra integral, hanya perlu melakukan perhitungan terhadap empat titik piksel. Perbedaan kecepatan ini akan semakin terlihat apabila fitur yang digunakan semakin besar, dan semakin banyak. Pada sebuah citra, jumlah fitur yang digunakan sangatlah banyak, jauh lebih banyak dari pada jumlah piksel pada citra tersebut, sehingga penggunaan citra integral dalam perhitungan nilai fitur *Haarlike* akan sangat membantu mempercepat proses perhitungan. Inilah alasan mengapa *Viola Jones* dapat mengevaluasi setiap fitur dengan sangat cepat.

Untuk memilih fitur *Haar* yang khusus untuk digunakan dalam proses pendeteksian wajah dan untuk menetapkan ambang batas maka digunakan teknik pembelajaran yang disebut sebagai algoritma *AdaBoost*.

4.1.3.3 Analisa Algoritma *AdaBoost* pada Proses Pendeteksian Wajah

Pada penelitian ini diterapkan algoritma *AdaBoost* yang bertujuan mengkombinasikan banyak citra-citra yang kurang tajam (*weak classifiers)* untuk menjadi citra-citra yang lebih tajam (*strong classifiers)* dengan memberi bobot kepada citra *weak classifiers.*

Proses tahapan algoritma *boosting* ini akan dijelaskan secara lebih terperinci berdasarkan gambar *flowchart* 4.6 berikut ini:



Gambar 4.7 *Flowchart* Algoritma *AdaBoost*

Sesuai dengan algoritma *AdaBoost,* cara kerja *AdaBoost* adalah sebagai berikut:

1. Terdapat $(x_1, y_1),\ldots (x_m, y_m)$; $x_i \in x$, $y_i \in \{-1, +1\}$

   M = 5 +ve (biru, kotak) sebagai sampel positif

   L = 5 –ve (merah, bulat) sebagai sampel negatif

   N = M + L = 10

   Inisialisasikan bobot $D_{(t=1)}(i)$= 1/10 untuk semua i = 1,2,..,10,

   jadi, $D_{(1)}(1) = 0.1$, $D_{(1)}(2) = 0.1,\ldots\ldots, D_{(1)}(10) = 0.1$

   $$h_t(x) = \begin{cases} 1 & \text{if } p_t f_t(x) < p_t \bar{} _t \\ 0 & otherwise \end{cases}$$

   Dimana $p_t = \text{polaritas}\{+1 \text{or} -1\}$, $\bar{} _t = v = treshold$

   f adalah fungsi : ($f = mu + c$), m,c merpakan konstanta, u,v merupakan variabel.

Sumbu paralel klasifier lemah

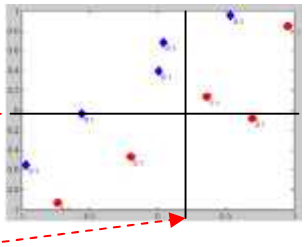
f adalah garis dari gradien m = 0 (garis horizontal)

posisi garis akan dikontrol oleh v_0

atau

f adalah garis gradien m = ∞ (garis vertikal)

posisi garis akan dikontrol oleh u_0



2. Untuk mengevaluasi bagaimana menghasilkan klasifier lemah yang telah diseleksi *h( )*, dilakukan evaluasi terhadap *error rate* dari klasifier lemah tersebut.

   ε_t = Peluang misklasifikasi *h( )*

   Periksa: If ε_t > = 0.5 (boleh jadi salah), maka training dihentikan

   - Karena dengan untuk mendefenisikan klasifier lemah harus lebih baik dari pilihan acak, maka peluangnya = 0.5
   - Jadiif ε_t > = 0.5 , dengan *h( )* adalah pilihan salah, bangun kembali *h"( )* dan lakukan training berdasarkan *h"( ).* Hasilnya:

3. Gunakan $\varepsilon_{t=1}$ = 0.3, karena 3 sampel menghasilkan klasifikasi yang salah.

$$£_t = \sum_{i=1}^{n} D_t(i) * I_{[h_t(x_i) \neq y_i]},$$

dimana $I_{[h_t(x_i) \neq y_i]} = \begin{cases} 1 \text{ if } [h_t(x_i) \neq y_i] \\ 0 \qquad otherwise \end{cases}$

$\varepsilon_{t=1} = 0.1 + 0.1 + 0.1 = 0.3$

Langkah 2 : $\sim_t = \frac{1}{2} \ln \frac{1-\varepsilon_t}{\varepsilon_t}$

dimana ε_t adalah pembobotan *error rate* dari klasifier h_t.

$\sim_{t=1} = \frac{1}{2} \ln \frac{1-0.3}{0.30} = 0.424$

4. Perbaharui bobot $D_t(i)$ untuk setiap *training sample i*

Langkah 3 : $D_{t+1}(i) = \frac{D_t(i) \exp(-\sim_t y_i h_t(x_i))}{Z_t}$

dimana Z_t = normalisasi faktor,

D_t = distribusi peluang

5. Temukan nilai Z awal (Normalisasi faktor)

$D_{t=1} = 0.1, \alpha_{t=1} = 0.424$

7 benar dan 3 sampel yang salah

t = 1

$Z_t = \sum_{y_i = h_i(x_i)} correct\_weight + \sum_{yi \neq h_i(x_i)} incorrect\_weight$

$Z_t = \sum_{y_i = h_i(x_i)} D_t(i) e^{-\alpha_t y_i h_t(x_i)} + \sum_{yi \neq h_i(x_i)} D_t(i) e^{-\alpha_t y_i h_t(x_i)} ----(i)$

klasifikasi benar : $y_i = h_i(x_i)$, jadi $y_i h_i(x_i) = +1$, masukkan kedalam *(i)*

klasifikasi salah : $y_i \neq h_i(x_i)$, jadi $y_i h_i(x_i) = -1$, masukkan kedalam *(i)*

$Z_t = \sum_{y_i = h_i(x_i)} D_t(i) e^{-\alpha_t (+1)} + \sum_{yi \neq h_i(x_i)} D_t(i) e^{-\alpha_t (-1)} = \sum_{y_i = h_i(x_i)} D_t(i) e^{-\alpha_t} + \sum_{yi \neq h_i(x_i)} D_t(i) e^{\alpha_t}$

$(total\_correct\_weight) + (total\_incorrect\_weight)$

$= 0.1 * 7 * e^{-0.424} + 0.1 * 3 * e^{+0.424} = 0.1 * 7 * 0.65 + 0.1 * 3 * 1.52$

$= 0.455 + 0.456$

$Z_{t=1} = 0.911$

6. Perbaharui D_t menjadi $D_{t+1,}$ jika klasifikasi benar, maka bobot D_{t+1} akan ditingkatkan.

$D_{t+1} = \frac{D_t(i)}{Z_t} e^{-\sim(correct)}$

$$= \frac{0.1}{Z_t} e^{-0.42}$$

$$D_{t+1}(i)_{correct} = \frac{0.1}{Z_t} 0.65$$

$$D_{t+1}(i)_{incorrect} = \frac{0.1}{Z_t} e^{-1} = \frac{0.1}{Z_t} e^{0.42} = 0.1 * 1.52$$

selama $Z_t = 0.911$, *So*

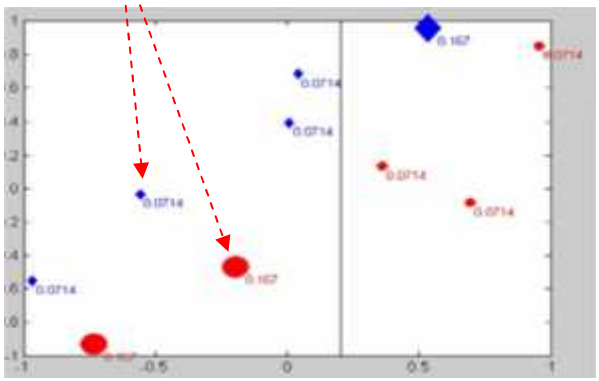
$$(\downarrow decrease) D_{t+1}(i)_{correct} = \frac{0.1}{0.911} e^{-1} = \frac{0.1}{0.911} 0.65 = 0.0714$$

$$(\uparrow increase) D_{t+1}(i)_{incorrect} = \frac{0.1}{0.911} e^{--1} = \frac{0.1}{0.911} 1.52 = 0.167$$

7. Kemudian lakukan *looping* terhadap *training* yang utama untuk yang kedua dengan t = 2.

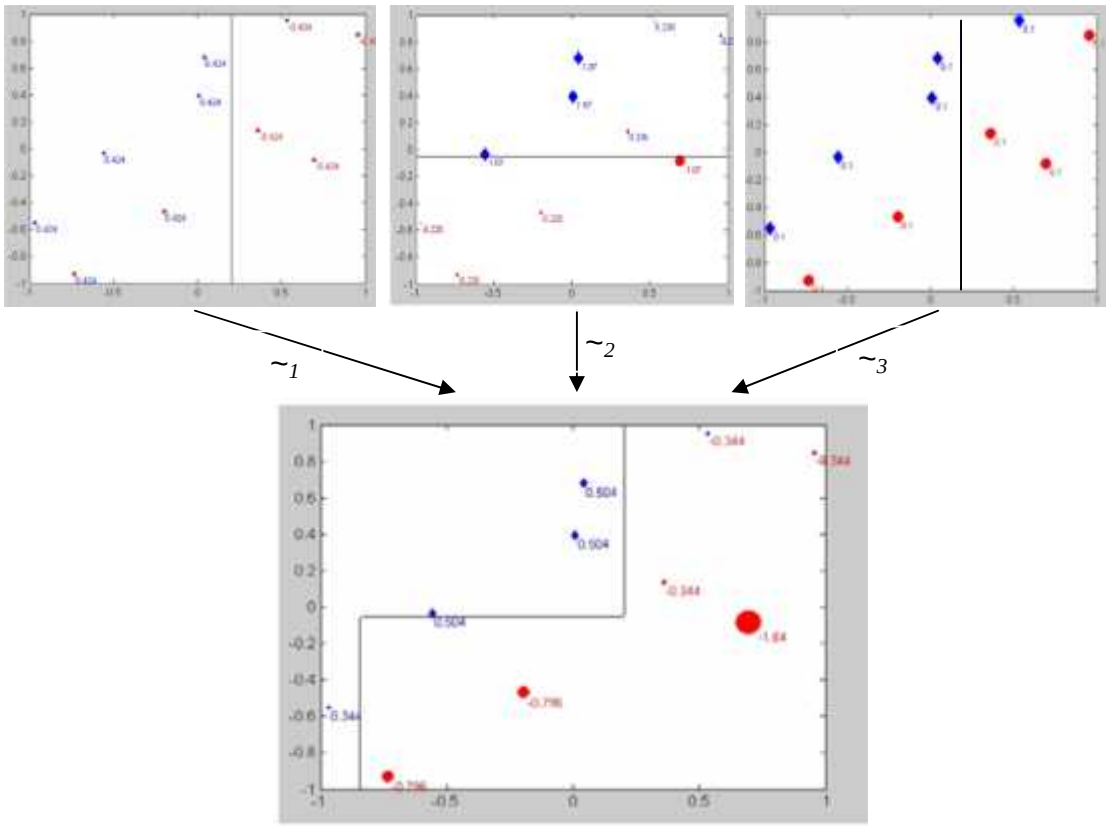
$$D_{t+1}(i)_{correct} = \frac{0.1}{0.911} e^{-1} = \frac{0.1}{0.911} 0.65 = 0.0714$$

$$D_{t+1}(i)_{incorrect} = \frac{0.1}{0.911} e^{--1} = \frac{0.1}{0.911} 1.52 = 0.167$$



8. Kemudian lakukan *looping* terhadap *training* yang utama untuk yang kedua dengan t = 2 dan ketiga dengan t = 3. Maka hasilnya dapat dilihat pada gambar berikut berdasarkan kombinasi yang dilakukan terhadap 3 klasifier lemah.

Kombinasi klasifier untuk t = 1,2,3



Gambar 4.8 Kombinasi Klasifier dengan t = 1,2,3

4.1.3.4 Analisa Metode *Cascaded Classifier* pada Proses Pendeteksian Wajah

Berdasarkan pada potongan *code* dalam *haarcascade_frontalface_alt.xml* yang telah dijelaskan pada bab 2, terdapat beberapa parameter dalam *template* yang digunakan dalam penelitian ini, yaitu terdapat 22 tahapan klasifikasi yang berupa *stage* 0 hingga *stage* 21. Jumlah tahapan ini menjukkan tingkatan dalam *classifier* dalam *haarcascade*. Kemudian pada awal bagian *code* program menunjukkan bahwa ukuran *window* yang akan digunakan dalam pengklasifikasian harus distandarkan menjadi berukuran 20 x 20, ini merupakan parameter untuk pengambilan sub-*region* dalam pembentukan fitur.

Setelah dikonversi ke dalam ukuran 20 x 20, selanjutnya setiap citra *sub-window* akan diklasifikasikan sebagai wajah ataupun bukan wajah, dengan memalui setiap tahapan yang ada di dalam *cascaded classifier* atau *template file* tersebut. Dari potongan *code* tersebut, dapat dilihat bahwa di dalam setiap tahapan

terdapat sebuah nilai *threshold* (*stage_threshold*) dan beberapa *tree*. *Tree* ini adalah *weak classifier* yang didapatkan dari setiap tahapan algoritma *boosting*. Jumlah *tree* di setiap tahapan berbeda-beda. Semakin tinggi tahapannya, maka semakin banyak pula jumlah *tree* yang digunakan. Contohnya, pada tahap 0 terdapat 3 *tree*, tahap 5 terdapat 4 *tree*, tahap 10 terdapat 80 *tree*, dan tahap 21 terdapat 213 *tree*. Jumlah *tree* pada suatu tahapan menunjukkan jumlah fitur yang digunakan untuk mengklasifikasi setiap *sub-window* yang mencapai tahapan tersebut. Bentuk fitur yang digunakan, bisa sama, tetapi bisa juga berbeda, begitu juga dengan posisi dan ukuran fitur tersebut. Namun tidak ada fitur yang persis sama (memiliki bentuk, posisi, dan ukuran yang sama) di *tree* yang berbeda.

Di setiap *tree* hanya terdapat satu *node*, yaitu *root node*. Di dalam setiap *node* ini, terdapat fitur *Haarlike* (*rects*), nilai *threshold*, serta nilai-nilai batasan minimum dan maksimum (*left_val* dan *right_val*) yang harus dipenuhi suatu fitur agar *sub-window* dapat lolos ke tahap selanjutnya. Untuk dapat melewati suatu tahapan, suatu *sub-window* harus berhasil melewati semua *tree* yang ada di dalam tahapan tersebut.

Perhitungan nilai fitur juga dilakukan di dalam *tree*. Berikut adalah penjelasan mengenai fitur *Haarlike* (*rects*) pada setiap *tree* berdasarkan file *xml* yang digunakan dalam penelitian ini.

```
<rects>
<_>3 7 14 4 -1.</_>
<_>3 9 14 2 2.</_>
</rects>
```

Angka pada kolom pertama dan ke dua menunjukkan posisi (x,y) fitur persegi pada *sub-window*. Angka pada kolom tiga menunjukkan lebar (*width*) fitur, dan angka pada kolom empat menunjukkan tinggi (*height*) fitur. Sedangkan angka pada kolom terakhir merupakan konstanta yang akan dikalikan dengan setiap nilai piksel pada fitur persegi tersebut. Apabila fitur persegi pada suatu baris bertemu dengan fitur persegi pada baris lain, maka setiap nilai pikselnya akan dikalikan dengan hasil penjumlahan antara angka terakhir dari masing-masing baris yang saling bertemu tersebut. Fitur persegi tersebut kemudian di-*threshold* dengan algoritma sebagai berikut:

```
IF piksel < 0 THEN set_to_black
ELSE set_to_white
```

Contoh: Angka-angka pada baris pertama, menunjukkan bahwa, fitur terletak pada piksel ke (3,7), dengan ukuran lebar x tinggi sebesar 14 x 4, dan setiap nilai piksel pada fitur tersebut akan dikalikan dengan -1. Sedangkan angka-angka pada baris ke dua menunjukkan bahwa, fitur terletak pada piksel ke (3,9), dengan ukuran 14 x 2, dan setiap nilai piksel pada fitur akan dikalikan dengan 2. Persegi pertama bertemu dengan persegi ke dua pada titik (3,9) hingga titik (16,10), sehingga setiap piksel pada daerah tersebut dikalikan dengan hasil penjumlahan angka terakhir dari baris pertama dan ke dua, yaitu (-1) + 2 = 1.

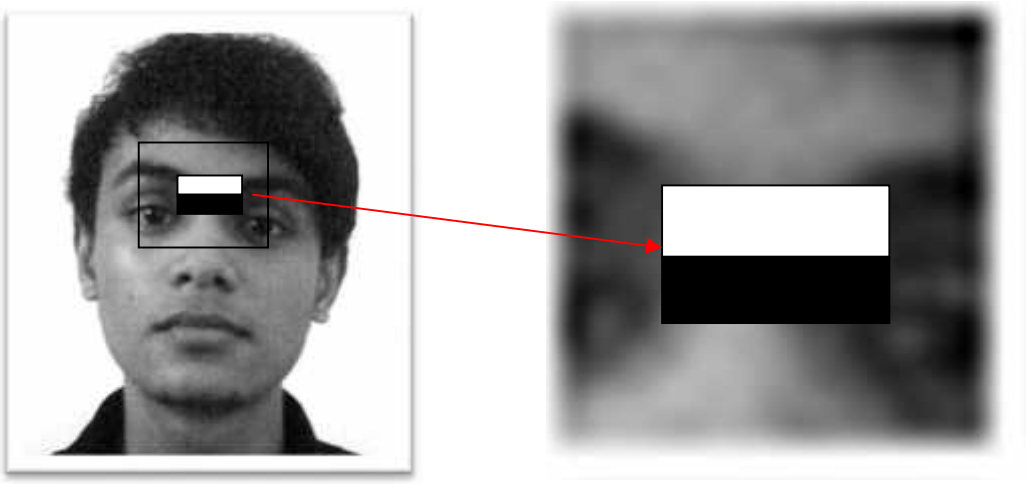
. Keberadaan ada atau tidaknya fitur wajah ditentukan dengan mengurangi nilai *pixel* di wilayah gelap dengan nilai *pixel* di wilayah terang. Jadi Setiap gambar dirubah kedalam warna hitam dan putih. Jika nilai dari hasil perbedaanya di atas dari ambang batas selama masa pembelajaran citra maka fitur tersebut dapat dikatakan ada.

Fitur-fitur tersebut merupakan gambaran dari wajah manusia yang dikelompokkan berdasarkan sisi yang terang dan sisi yang gelap seperti yang dapat dilihat pada Gambar 4.7 dibawah ini. Daerah mata memiliki sisi yang lebih gelap daripada bagian di antara dua mata.

Sebagai simulasi dicontohkan terdapat sebuah citra yang akan dideteksi dilakukan proses sebagai berikut:

Gambar 4.9 Contoh Citra Simulasi

1. Pengambilan sub-*region* atau sub-*window* yang telah dilakukan *scaling* menjadi berukuran 20 x 20 piksel.



Gambar 4.10 Pengambilan Sub-*window* yang telah di-*scaling* 20 x 20 Piksel

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	2	4	3	5	2	6	8	5	8	9	9	4	4	5	6	5	6	5	7	8
2	4	3	5	6	7	4	6	10	4	3	11	3	12	4	2	5	34	6	7	4
3	6	7	4	5	5	5	6	6	7	7	7	7	8	8	9	9	7	8	8	9
4	6	7	4	5	5	5	6	6	7	7	7	7	8	8	9	9	8	5	5	8
5	6	7	4	5	5	5	6	6	7	7	7	7	8	8	9	9	8	6	21	7
6	6	7	4	5	5	5	6	6	7	7	7	7	8	8	9	9	7	7	22	7
7	6	7	4	5	5	5	6	6	7	7	7	7	8	8	9	9	8	8	6	8
8	5	7	6	5	5	3	2	5	3	12	3	5	5	5	5	6	4	3	4	8
9	7	3	1	7	3	5	4	2	6	4	7	8	9	8	2	2	4	7	7	8
10	6	4	2	1	4	4	6	3	7	8	5	6	7	4	6	5	5	5	3	8
11	5	5	5	5	5	5	7	4	2	6	7	8	5	9	7	4	6	7	2	9
12	4	3	2	1	7	2	3	6	4	3	5	6	4	5	6	3	7	7	5	9
13	8	4	23	14	5	7	9	8	7	5	5	5	8	7	7	4	7	7	7	6
14	9	6	3	0	7	5	7	5	6	3	6	8	7	9	3	23	13	4	6	4
15	15	7	1	9	4	6	2	7	8	6	7	7	8	9	9	4	7	8	5	5
16	6	8	10	12	4	6	2	7	8	6	7	7	8	9	9	3	7	7	6	4
17	5	6	7	8	4	6	2	7	8	6	7	7	8	9	9	2	8	8	45	4
18	8	8	8	8	4	6	2	7	8	6	7	7	8	9	9	2	8	8	6	4
19	9	7	5	3	3	4	6	5	7	8	5	4	7	8	8	1	8	8	5	3
20	9	8	7	6	4	5	3	6	4	5	5	5	6	7	7	8	9	8	4	2

Gambar 4.11 Nilai Asli pada Citra Simulasi Sub-*window*

2. Misalkan setelah diambil sub-*window* berukuran 20 x 20 piksel, maka diperoleh nilai pikselnya seperti gambar 4.10.
3. Kemudian dilakukan perhitungan citra integral, hasilnya adalah seperti gambar 4.11 berikut:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	2	8	9	10	17	23	31	36	44	53	62	66	70	75	81	86	92	97	104	112
2	6	13	21	32	41	51	65	80	92	104	124	131	147	156	164	174	214	225	239	251
3	12	26	38	54	68	83	103	124	143	162	189	203	227	244	261	280	327	346	368	389
4	18	39	55	76	95	115	141	168	194	220	254	275	307	332	358	386	441	465	492	521
5	24	52	72	98	122	147	81	212	245	278	319	347	387	420	455	492	555	585	633	669
6	30	65	89	120	149	179	217	256	296	336	384	419	467	508	552	598	668	705	775	818
7	36	78	106	142	176	211	255	300	347	394	449	491	547	596	649	704	782	827	903	954
8	41	91	111	315	204	242	288	338	388	447	505	552	613	667	725	786	868	916	996	1055
9	48	142	135	183	225	268	318	370	426	489	554	609	679	741	801	864	950	1005	1092	1159
10	54	200	213	196	242	289	345	400	463	534	604	665	742	808	874	942	1033	1093	1183	1258
11	59	122	162	216	267	319	382	441	506	583	660	729	811	886	959	1031	1128	1195	1287	1371
12	63	188	197	226	284	338	404	469	538	618	700	775	861	941	1020	1095	1199	1273	1370	1463
13	71	263	295	275	338	399	474	547	623	708	795	875	969	1056	1142	1221	1332	1413	1517	1616
14	80	161	201	293	363	429	511	589	671	759	852	940	1041	1137	1226	1328	1452	1537	1647	1750
15	95	263	290	325	399	471	555	640	730	824	924	1019	1128	1233	1331	1437	1568	1661	1776	1884
16	101	372	409	361	439	517	603	695	793	893	1000	1102	1219	1333	1440	1549	1687	1787	1908	2020
17	106	221	258	387	469	553	641	740	846	952	1066	1175	1300	1423	1539	1650	1796	1904	2070	2186
18	114	343	368	419	505	595	685	791	905	1017	1138	1254	1387	1519	1644	1757	1911	2027	2199	2319
19	123	473	503	443	532	626	722	833	954	1074	1200	1320	1460	1600	1733	1847	2009	2133	2310	2433
20	132	270	291	473	566	665	764	881	1006	1131	1262	1387	1533	1680	1820	1942	2113	2245	2426	2551

Gambar 4.12 Hasil Pengintegralan Citra

4. Kemudian dikalkulasikan nilai *rect* berdasarkan file xml.

```
<rects>
<_>3 7 14 4 -1.</_>
<_>3 9 14 2 2.</_>
</rects>
```

Sehingga hasilnya terbentuk model fitur persegi yang terdiri dari bagian hitam dan putih, untuk lebih jelasnya dapat dilihat pada gambar berikut ini:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	2	8	9	10	17	23	31	36	44	53	62	66	70	75	81	86	92	97	104	112
2	6	13	21	32	41	51	65	80	92	104	124	131	147	156	164	174	214	225	239	251
3	12	26	38	54	68	83	103	124	143	162	189	203	227	244	261	280	327	346	368	389
4	18	39	55	76	95	115	141	168	194	220	254	275	307	332	358	386	441	465	492	521
5	24	52	72	98	122	147	81	212	245	278	319	347	387	420	455	492	555	585	633	669
6	30	65	89	120	149	179	217	256	296	336	384	419	467	508	552	598	668	705	775	818
7	36	78	106	142	176	211	255	300	347	394	449	491	547	596	649	704	782	827	903	954
8	41	91	111	315	204	242	288	338	388	447	505	552	613	667	725	786	868	916	996	1055
9	48	100	135	183	225	268	318	370	426	489	554	609	679	741	801	864	950	1005	1092	1159
10	54	110	147	196	242	289	345	400	463	534	604	665	742	808	874	942	1033	1093	1183	1258
11	59	120	162	216	267	319	382	441	506	583	660	729	811	886	959	1031	1128	1195	1287	1371
12	63	127	171	226	284	338	404	469	538	618	700	775	861	941	1020	1095	1199	1273	1370	1463
13	71	139	206	275	338	399	474	547	623	708	795	875	969	1056	1142	1221	1332	1413	1517	1616
14	80	154	224	293	363	429	511	589	671	759	852	940	1041	1137	1226	1328	1452	1537	1647	1750
15	95	176	247	325	399	471	555	640	730	824	924	1019	1128	1233	1331	1437	1568	1661	1776	1884
16	101	190	271	361	439	517	603	695	793	893	1000	1102	1219	1333	1440	1549	1687	1787	1908	2020
17	106	201	289	387	469	553	641	740	846	952	1066	1175	1300	1423	1539	1650	1796	1904	2070	2186
18	114	217	313	419	505	595	685	791	905	1017	1138	1254	1387	1519	1644	1757	1911	2027	2199	2319
19	123	233	334	443	532	626	722	833	954	1074	1200	1320	1460	1600	1733	1847	2009	2133	2310	2433
20	132	250	358	473	566	665	764	881	1006	1131	1262	1387	1533	1680	1820	1942	2113	2245	2426	2551

Gambar 4.13 Hasil Pembentukan Fitur

5. Kemudian dilakukan kalkulasi untuk mendapatkan nilai fitur berdasarkan gambar diatas. Prosesnya adalah:

   Nilai Fitur = | (total piksel hitam) – (total piksel putih) |

   = | {(106 + 950) – (135 + 782)} – (135+1128) – (162 + 950) |

   = | 139 – 151 |

   = 12

   Nilai Fitur ini kemudian dilakukan proses *tresholding* dengan memberikan pembobotan berdasarkan algoritma *AdaBoost.*

4.1.4 Analisa Metode *Face Recognition* dengan Metode *Eigenface*

Proses pengenalan wajah diawali dengan pengambilan citra wajah yang nantinya akan dikenali oleh sistem dan dijadikan citra referensi. Kemudian tahap selanjutnya adalah pengumpulan citra wajah dalam sebuah folder citra untuk dihitung nilai *eigen*-nya. Setelah nilai *eigen* didapatkan kemudian tahap selanjutnya yaitu mengambil citra wajah yang akan diujikan ke sistem pengenalan wajah, citra uji ini terlebih dahulu dihitung juga nilai *eigen*-nya. Kemudian dengan menggunakan metode *eucledian distance* akan ditelusuri citra uji tersebut apakah dapat dikenali oleh sistem. Caranya adalah dengan membandingkan nilai *eigen* dari citra uji dengan semua citra referensi yang berada dalam sistem. Kemudian hasil kalkulasi *distance similarity* menggunakan *eucledien distance* yang paling kecil diasumsikan sebagai citra yang paling mirip. Untuk lebih memahami proses kerja metode *eigenface* ini hingga mampu mengenali citra wajah manusia akan dijelaskan pada sub bab berikut ini.

4.1.4.1 Ekstraksi Ciri Citra Referensi

Ekstraksi ciri citra referensi merupakan sebuah proses yang bertujuan untuk mendapatkan nilai ciri dari suatu citra, dalam hal ini adalah citra wajah yang menjadi referensi. Untuk lebih mudah memahami rumus perhitungan dalam ekstraksi ciri citra ini, diberikan contoh 3 buah citra dengan ukuran 2 x 2 *pixel*. Perhitungan ini hanya untuk gambaran umum dari rumus-rumus yang ada. Berikut ini adalah contoh tahapan proses ekstraksi ciri citra referensi dengan menggunakan metode *eigenface*:

1. Matrix 2 x 2 diasumsikan sebagai suatu citra sederhana yang setiap elemennya dianggap sebagai nilai pixel dari citra. Misal terdapat citra A, citra B dan citra C. Sehingga diketahui nilai M = 3, M adalah jumlah citra referensi.

   $$A = \begin{matrix} 7 & 4 \\ 2 & 8 \end{matrix} \qquad B = \begin{matrix} 9 & 1 \\ 5 & 4 \end{matrix} \qquad C = \begin{matrix} 3 & 7 \\ 2 & 6 \end{matrix}$$

2. Kemudian masing-masing citra dirubah menjadi matrix baris

A = 7 4 2 8

B = 9 1 5 4

C = 3 7 2 6

3. Selanjutnya seluruh citra digabung menjadi satu matriks lm dengan ukuran M (M=3) baris dan 4 (2x2) kolom.

$$lm = \begin{matrix} 7 & 4 & 2 & 8 \\ 9 & 1 & 5 & 4 \\ 3 & 7 & 2 & 6 \end{matrix}$$

4. Setelah itu dilakukan proses mencari nilai *mean* atau rata-rata sesuai persamaan 2.8. setiap kolom dari *matrix* lm akan dihitung nilai rataannya.

$$\text{Kolom 1} = \begin{matrix} 7 \\ 9 \\ 3 \end{matrix} = \frac{1}{3}(7+9+3) = 6 \text{ (dilakukan pembulatan)}$$

$$\text{Kolom 2} = \begin{matrix} 4 \\ 1 \\ 7 \end{matrix} = \frac{1}{3}(4+1+7) = 4$$

$$\text{Kolom 3} = \begin{matrix} 2 \\ 5 \\ 2 \end{matrix} = \frac{1}{3}(2+5+2) = 3$$

$$\text{Kolom 4} = \begin{matrix} 8 \\ 4 \\ 6 \end{matrix} = \frac{1}{3}(8+4+6) = 6$$

Sehingga diperoleh sebuah matriks baru, yaitu mean = 6 4 3 6

5. Kemudian mencari ciri setiap citra referensi sesuai persamaan 2.9, prosesnya adalah sebagai berikut:

$$\text{Fitur} = \begin{matrix} 7-6 & 4-4 & 2-3 & 8-6 \\ 9-6 & 1-4 & 5-3 & 4-6 \\ 3-6 & 7-4 & 2-3 & 6-6 \end{matrix}$$

$$= \begin{matrix} 1 & 0 & -1 & 2 \\ 3 & -3 & 2 & -2 \\ -3 & 3 & -1 & 0 \end{matrix}$$

Untuk menyederhanakan komputasi, maka setiap nilai matriks yang bernilai negative dijadikan 0. Sehingga nilai fitur menjadi:

$$\text{Fitur} = \begin{matrix} 1 & 0 & 0 & 2 \\ 3 & 0 & 2 & 0 \\ 0 & 3 & 0 & 0 \end{matrix}$$

6. Selanjutnya adalah mencari mattriks kovarian berdasarkan pada rumus 2.10. prosesnya adalah sebagai berikut:

   $C = (\text{fitur x})(\text{fitur}^T)$

$$C = \begin{matrix} 1 & 0 & 0 & 2 \\ 3 & 0 & 2 & 0 \\ 0 & 3 & 0 & 0 \end{matrix} \text{ x } \begin{matrix} 1 & 3 & 0 \\ 0 & 0 & 3 \\ 0 & 2 & 0 \\ 2 & 0 & 0 \end{matrix}$$

$$= \begin{matrix} 5 & 3 & 0 \\ 3 & 13 & 0 \\ 0 & 0 & 9 \end{matrix}$$

7. Setelah ditemukan nilai matriks kovarian, maka selanjutnya adalah mencari nilai eigen dan vector eigen. Proses ini berdasarkan pada persamaan 2.11. Berikut penjabaran prosesnya:

   a. Nilai *eigen* didapatkan dengan persamaan:

      $\text{Det}[\lambda I - C] = 0$

$$\det \begin{matrix} \lambda - 5 & -3 & 0 \\ -3 & \lambda - 13 & 0 \\ 0 & 0 & \lambda - 9 \end{matrix} = 0$$

      sehingga diperoleh persamaan karakter:

      $(\lambda - 9)\{(\lambda - 5)(\lambda - 13) - 9\} = 0$

      $(\lambda - 9)\{\lambda^2 - 18\lambda + 56\} = 0$

      $(\lambda - 9)(\lambda - 4)(\lambda - 14) = 0$

      Maka nilai $\lambda_{1,2,3}$ adalah:

      $\lambda - 4 = 0$

      $\lambda = 4$

      $\lambda - 9 = 0$

      $\lambda = 9$

      $\lambda - 14 = 0$

$\lambda = 14$

Maka nilai *eigen*-nya diperoleh adalah:
$$\begin{matrix} 4 & 0 & 0 \\ 0 & 9 & 0 \\ 0 & 0 & 14 \end{matrix}$$

b. Kemudian dicari *vector eigen* berdasarkan pada persamaan 2.12. Prosesnya adalah sebagai berikut:

$AX = \lambda X$

$$\begin{matrix} 5-\lambda & -3 & 0 \\ -3 & 13-\lambda & 0 \\ 0 & 0 & 9-\lambda \end{matrix} \begin{matrix} x1 \\ x2 \\ x3 \end{matrix} = \begin{matrix} 0 \\ 0 \\ 0 \end{matrix}$$

Untuk $\lambda = 4$, persamaan linearnya adalah sebagai berikut:

$x_1 + 3x_2 + 0 = 0$

$3x_1 + 9x_2 + 0 = 0$

$0 + 0 + 5x_3 = 0$

$x_1 = -3x_2$

$x_3 = 0$

Vector eigen yang sesuai adalah $x = \begin{matrix} -3x2 \\ x2 \\ 0 \end{matrix}$ misalkan $x_2 = t$, maka

$$x = \begin{matrix} -3t \\ t \\ 0 \end{matrix}$$

Jika proses diatas dilanjutkan dengan $\lambda = 9$ dab $\lambda = 14$, maka diperoleh *vector eigen* sebagai berikut :

$$\lambda = 9, x = \begin{matrix} 0 \\ 0 \\ 1 \end{matrix}$$

$$\lambda = 14, x = \begin{matrix} t \\ 3t \\ 0 \end{matrix}$$

Untuk penyelesaian diambil nilai t = 0.3, maka didapatkan *eigen vector* sebagai berikut:

$$\begin{matrix} -0,9 & 0 & 0,3 \\ 0,3 & 9 & 0,9 \\ 0 & 1` & 0 \end{matrix}$$

8. Selanjutnya adalah mencari *eigenface* berdasarkan persamaan 2.13, prosesnya adalah sebagai berikut:

   Ef = fitur^T x *vector eigen*

$$Ef = \begin{matrix} 1 & 3 & 0 \\ 0 & 0 & 3 \\ 0 & 2 & 0 \\ 2 & 0 & 0 \end{matrix} \text{ x } \begin{matrix} -0{,}9 & 0 & 0{,}3 \\ 0{,}3 & 0 & 0{,}9 \\ 0 & 1` & 0 \end{matrix}$$

$$Ef = \begin{matrix} -0{,}9 + 0{,}9 + 0 & 0 + 0 + 0 & 0{,}3 + 2{,}7 + 0 \\ 0 + 0 + 0 & 0 + 0 + 3 & 0 + 0 + 0 \\ 0 + 0{,}6 + 0 & 0 + 0 + 0 & 0{,}6 + 0 + 0 \\ -1{,}8 + 0 + 0 & 0 + 0 + 0 & 0{,}6 + 0 + 0 \end{matrix}$$

$$Ef = \begin{matrix} 0 & 0 & 3 \\ 0 & 3 & 0 \\ 0{,}6 & 0 & 1{,}8 \\ -1{,}8 & 0 & 0{,}6 \end{matrix}$$

   Berdasarkan matriks *eigenface* yang diperoleh diatas, dapat dipahami bahwa setiap kolom pada matriks tersebut semakin kekanan nilainya semakin besar. Misalkan nilai terbesar diambil dua kolom dari yang paling kanan, sehingga diketahui N = 2, N adalah nilai *eigenface* terbesar, maka diperoleh matriks sebagai berikut:

$$EfN = \begin{matrix} 3 & 0 \\ 0 & 3 \\ 1{,}8 & 0 \\ 0{,}6 & 0 \end{matrix}$$

9. Tahap terakhir adalah mencari nilai bobot dari masing-masing citra referensi berdasarkan pada persamaan 2.14, prosesnya yaitu;

   W = fitur x EfN

$$W = \begin{matrix} 1 & 0 & 0 & 2 \\ 3 & 0 & 2 & 0 \\ 0 & 3 & 0 & 0 \end{matrix} \text{ x } \begin{matrix} 3 & 0 \\ 0 & 3 \\ 1{,}8 & 0 \\ 0{,}6 & 0 \end{matrix}$$

$$W = \begin{matrix} 3 + 0 + 0 + 1{,}2 & 0 + 0 + 0 + 0 \\ 9 + 0 + 3{,}6 + 0 & 0 + 0 + 0 + 0 \\ 0 + 0 + 0 + 0 & 0 + 9 + 0 + 0 \end{matrix}$$

$$W = \begin{matrix} 4{,}2 & 0 \\ 12{,}6 & 0 \\ 0 & 9 \end{matrix}$$

Keterangan: W = bobot dari suatu citra referensi

Baris I = bobot citra referensi A

Baris II = bobot citra referensi B

Baris III = bobot citra referensi C

4.1.4.2 Ekstraksi Ciri Citra Uji

Tahapan proses dalam ekstraksi ciri citra uji proses awalnya sama dengan proses pencarian *eigenface* pada citra referensi, perbedaanya hanya pada proses pengujian antara kedua citra tersebut. Tahapan dalam ekstraksi ciri citra uji dijelaskan sebagai berikut:

1. Misalnya disimulasikan untuk proses pengujian diketahui citra uji adalah: citra x = $\begin{matrix} 8 & 2 \\ 5 & 5 \end{matrix}$
2. Lakukan konversi matriks citra uji menjadi 1D

   X = [8 2 5 5]
3. Selanjutnya pencarian fitur, prosesnya sebagai berikut:

   Fitur x = x – mean (citra referensi)

   Fitur x = [8 2 5 5] - [6 4 3 6]

   Fitur x = [8-6 2-4 5-3 5-6]

   Fitur x = [2 -2 2 -1]

   Kemudian lakukan pembulatan positif untuk mempermudah komputasi selanjutnya, sehingga menjadi:

   Fitur x = [2 0 2 0]
4. Selanjutnya, cari nilai bobot citra uji x, berikut ini prosesnya:

   Wx = (fitur x)(EfN)

   $Wx = [2\ 0\ 2\ 0] \times \begin{matrix} 3 & 0 \\ 0 & 3 \\ 1{,}8 & 0 \\ 0{,}6 & 0 \end{matrix}$

   Wx = [6+0+3 6+0 0+0+0+0]

Wx = [9,6 0]

Hasil dari matriks Wx yang ditemukan, menunjukkan proses ekstraksi ciri citra uji selesai dilakukan.

4.1.4.3 Proses *Recognition* dengan Metode *Eucledien Distance*

Berdasarkan kalkulasi yang dilakukan pada perhitungan diatas yang menghasilkan nilai bobot dari citra yang sebagai referensi dan citra uji, yaitu matriks W dan Wx. Proses penghitungan jarak kemiripan ini berdasarkan pada persamaan 2.16. Citra yang memiliki nilai jarak yang terdekat diasumsikan sebagai wajah yang mirip dari citra yang diuji.

Proses perbandinganan untuk mencari jarak paling dekat antara masing-masing citra referensi dengan citra uji dilakukan dengan membandingkan nilai bobot dari matriks W terhadap matriks Wx. Berikut proses perhitungannya berdasarkan konsep *eucledien distance*:

Diketahui bahwa nilai matriks W = $\begin{matrix} 4{,}2 & 0 \\ 12{,}6 & 0 \\ 0 & 9 \end{matrix}$ dan Wx = Wx = [9,6 0]

Jarak pertama (D_1) adalah jarak antara citra uji X terhadap citra referensi A.

$D_1 = \sqrt{(9{,}6 - 4{,}2)^2 + (0 - 0)^2} = 5{,}4$

Jarak pertama (D_2) adalah jarak antara citra uji X terhadap citra referensi B.

$D_2 = \sqrt{(9{,}6 - 12{,}6)^2 + (0 - 0)^2} = 3$

Jarak pertama (D_3) adalah jarak antara citra uji X terhadap citra referensi C.

$D_3 = \sqrt{(9{,}6 - 0)^2 + (0 - 9)^2} = 13{,}15$

Berdasarkan hasil perhitungan dari masing-masing citra diatas, maka dapat dilihat bahwa nilai jarak *eucledien* terkecil adalah 3. Maka diasumsikan nilai citra uji yang dibandingkan ini adalah sebagai citra yang mirip, yaitu citra B. Untuk membuktikan kemiripannya dapat lansung dilihat dari masing-masing matriks citranya:

x = $\begin{matrix} 8 & 2 \\ 5 & 5 \end{matrix}$ B = $\begin{matrix} 9 & 1 \\ 5 & 4 \end{matrix}$

Kedua matriks tersebut memiliki elemen yang nilainya berdekatan dibandingkan dengan matriks lainnya, maka matriks tersebut diasumsikan yang citra yang mirip.

4.1.5 Analisa Kebutuhan Perangkat Lunak

Spesifikasi kebutuhan perangkat lunak dibagi menjadi dua bagian yaitu SKPL-F (Spesifikasi kebutuhan perangkat lunak fungsional) dan SKPL-NF (Spesifikasi kebutuhan perangkat lunak non-fungsional). Berikut ini adalah tabel dari spesifikasi kebutuhan perangkat lunak pengenalan wajah :

Tabel 4.1 Spesifikasi Kebutuhan Perangkat Lunak Fungsioanal

Kode	**Kebutuhan**
SKPL-F001	Perangkat lunak harus mampu mendeteksi areal wajah pada citra yang mengandung wajah.
SKPL-F002	Perangkat lunak harus dapat mengenali gambar wajah.
SKPL-F003	Perangkat lunak harus mampu mengukur tingkat akurasideteksi wajah dengan memberikan gamabar berbagai kondisi pada metode *Viola Jones*.
SKPL-F004	Perangkat lunak harus dapat mengolah data wajah Untuk proses pengenalan wajah menggunakan metode *eigenface*.

Tabel 4.2 Spesifikasi Kebutuhan Perangkat Lunak Non-Fungsioanal

Kode	**Kebutuhan**
SKPL-NF001	Pengguna perangkat lunak menggunakan wajah sebagai media untuk dideteksi oleh perangkat lunak.
SKPL-NF002	Kondisi lingkungan pengguna harus dalam keadaan bercahaya.
SKPL-NF003	Perangkat keras yang digunakan adalah kamera dan komputer dengan spesifikasi yang cukup baik dalam proses pengolahan citra.
SKPL-NF004	Perangkat lunak ini menggunakan sistem operasi Windows.

4.1.6 Analisa Kebutuhan Non-Fungsional

Analisis kebutuhan non-fungsional ini berfungsi menjelaskan pendukung sistem yang akan dijalankan. Adapun kebutuhan-kebutuhan yang diperlukan untuk mendukung kinerja dari perangkat lunak yaitu **:**

1. Analisis Kebutuhan Perangkat Keras

   Komputer merupakan gabungan dari perangkat keras dan perangkat lunak yang satu sama lain saling mendukung kinerjanya. Perangkat lunak memerintahkan perangkat keras instruksi untuk melakukan tugas-tugas tertentu. Pada perangkat lunak *face detection* dan *recognition* ini digunakan sebuah komputer berbasis *desktop*. Adapun kriteria dari komputer yang dibutuhkan saat pembangunan maupun menjalankannya perangkat lunak ini adalah :

   Tabel 4.3 Kebutuhan Perangkat Keras

No.	**Perangkat Keras**	**Spesifikasi**
1	Processor	AMD Dual-Core C-50
2	Monitor	Acer 11,6”
3	VGA	AMD Radeon HD 6250 Graphics
4	Memory	2 GB
5	Kamera	*External Webcam Lexmark X422, resolution up to* 5 *megapixel*

2. Analisis Kebutuhan Perangkat Lunak

   Perangkat lunak digunakan untuk memberikan perintah-perintah kepada perangkat keras untuk melakukan instruksi-instruksi. Berikut kriteria dari perangkat lunak yang dibutuhkan untuk mendukung perangkat lunak *face detection* dan *recognition* ini adalah:

   a. Sistem operasi : *Windows 7 Ultimate 32-bit Operating System*
   b. Bahasa pemrograman : *C#* (*C Sharp*) dan xml
   c. *Editor/compiler* : *Microsoft Visual Studio 2010 Express*
   d. *Protyping* : *Microsoft Office Visio 2007*
   e. Perancangan : *Notepad++* dan *Microsoft Visual C# 2010 Express*

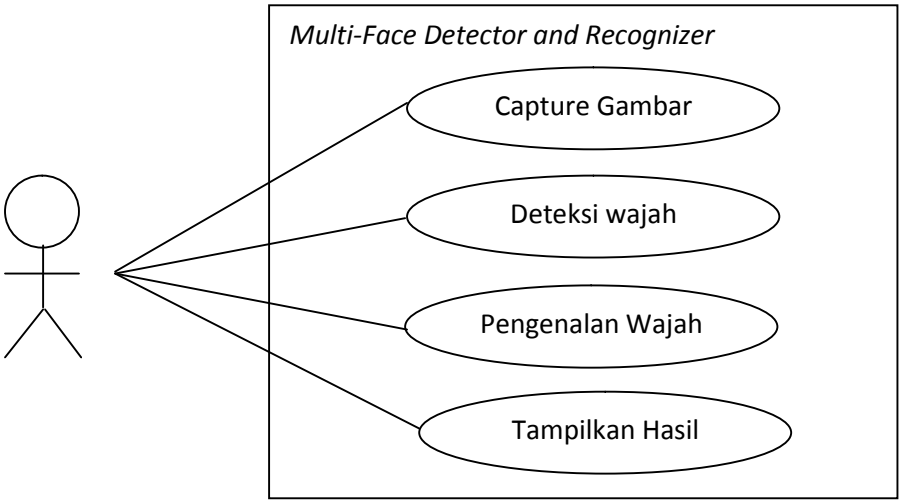
4.1.7 Analisa Kebutuhan Fungsional

Analisa kebutuhan fungsional merupakan alur dan pekerjaan dari perangkat lunak yang akan dibangun. Dalam perangkat lunak pendeteksian dan

pengenalan wajah ini digunakan metode pendekatan berorientasi objek.

Dalam penelitian ini penulis penulis merancang analisa kebutuhan fungsional berupa: *use case diagram*, *use case scenario* dan *activity diagram*. Berikut ini dijelaskan secara detil mengenai masing-masing perancangan berdasarkan pendekatan berorientasi objek:

1. *Use Case Diagram*



Gambar 4.14 *Use Case Diagram*

Pada gambar di atas terdapat aktor yang merupakan pengguna yang berhubungan langsung dengan perangkat lunak. *Setiap use case* dijelaskan dalam *use case scenario*.

2. *Use case scenario*

Nama *use case* : *Capture* gambar

Aktor : Pengguna dan kamera

Trigger : Pengguna menjalankan aplikasi pendeteksi dan pengenalan wajah

Skenario :

Tabel 4.4 Skenario *Capture* Gambar

No	Aksi Aktor Pada Skenario Normal	No	Aksi Sistem Pada Skenario Normal
1	Pengguna menjalankan aplikasi pendeteksian dan pengenalan wajah		
		2	Aplikasi menampilkan form utama aplikasi
3	Pengguna menekan tombol deteksi dan kenali wajah		
		4	Aplikasi menampilkan layar kamera pendeteksian dan pengenalan wajah

Nama *use case* : Deteksi Wajah

Aktor : Pengguna dan kamera

Trigger : Pengguna menjalankan menekan tombol deteksi dan kenali wajah

Skenario :

Tabel 4.5 Skenario Deteksi Wajah

No	Aksi Aktor Pada Skenario Normal	No	Aksi Sistem Pada Skenario Normal
1	Pengguna menekan tombol kenali dan deteksi wajah		
		2	Aplikasi menampilkan layar pendeteksian dan pengenalan wajah
3	Pengguna menghadapkan sebuah atau beberapa wajah didepen kamera		
		4	Aplikasi menampilkan kotak areal wajahpada setiap wajah yang terdeteksi kamera

Nama *use case* : Pengenalan Wajah

Aktor : Pengguna dan kamera

Trigger : Pengguna menjalankan menekan tombol deteksi dan kenali wajah

Skenario :

Tabel 4.6 Skenario Pengenalan Wajah

No	**Aksi Aktor Pada Skenario Normal**	**No**	**Aksi Sistem Pada Skenario Normal**
1	Pengguna menekan tombol Kenali dan deteksi wajah		
		2	Aplikasi menampilkan Layar pendeteksian dan pengenalan wajah
3	Pengguna menghadapkan sebuah Atau beberapa wajah didepen kamera		
		4	Aplikasi menampilkan kotak areal wajah pada setiap wajah yang terdeteksi kamera
5	Pengguna menekan tombol “daftarkan wajah”		
		6	Aplikasi akan memberikan pesan wajah telah terdaftar dan tersimpan dalam database.

Nama *use case* : Menampilkan Hasil Deteksi dan pengenalan wajah

Aktor : Pengguna dan kamera

Trigger : Pengguna menjalankan menekan tombol deteksi dan kenali wajah

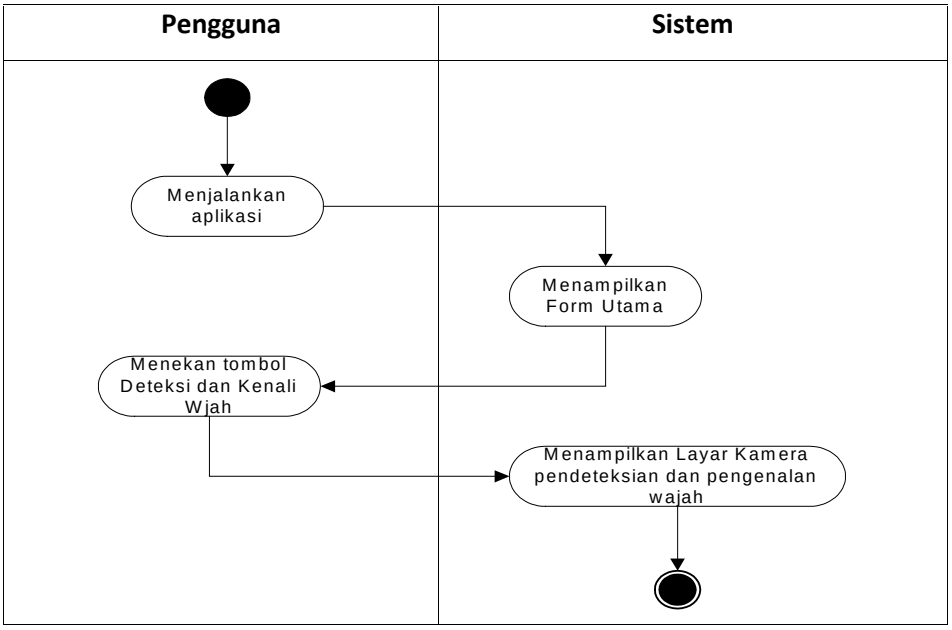
Skenario :

Tabel 4.7 Skenario Menampilkan Hasil Deteksi dan Pengenalan Wajah

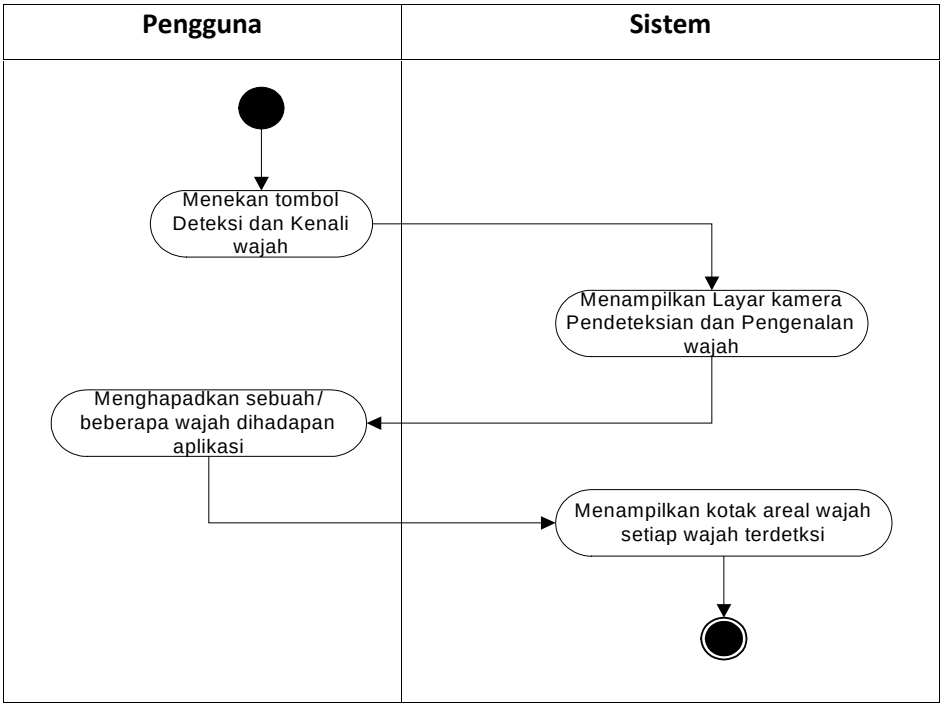
No	Aksi Aktor Pada Skenario Normal	No	Aksi Sistem Pada Skenario Normal
1	Pengguna menekan tombol kenali dan deteksi wajah		
		2	Aplikasi menampilkan Layar pendeteksian dan pengenalan wajah
3	Pengguna menghadapkan sebuah atau beberapa wajah didepen kamera		
		4	Pada kotak hasil aplikasi akan menampilkan informasi wajah berupa nama pemilik wajah yang telah didaftarkan (jika dikenali)
		5	Aplikasi memberikan informasi jumlah wajah yang terdeteksi atau dikenali
		6	Aplikasi secara lansung menampilkan nama pemilik wajah pada kotak areal wajah

3. *Activity* Diagram

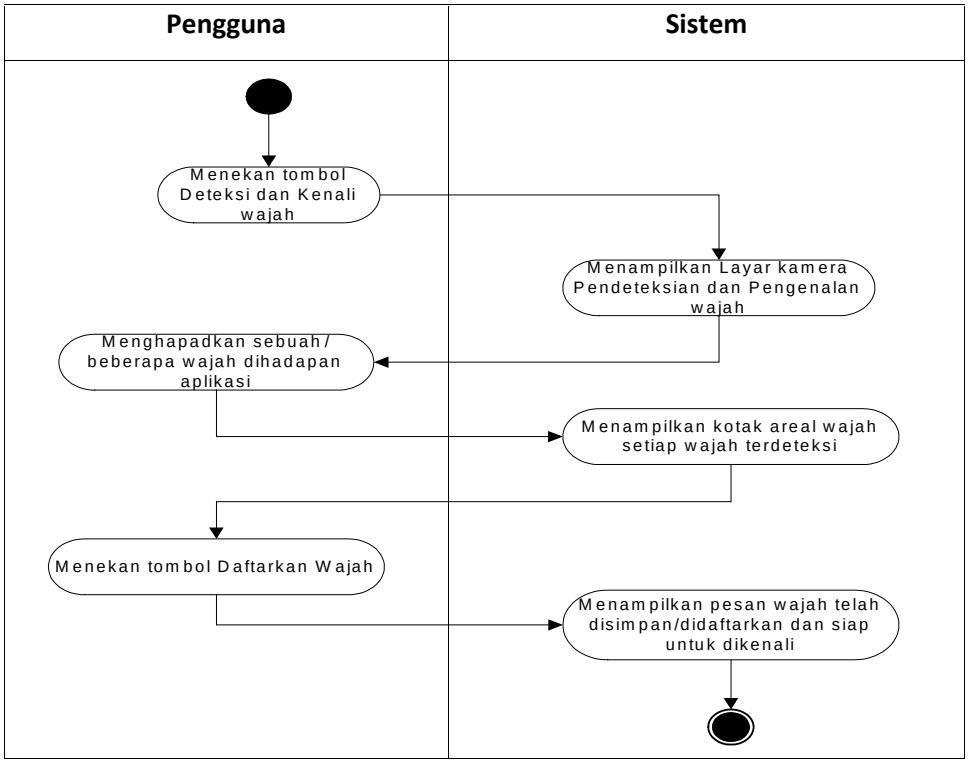
   Untuk menjelaskan aktifitas setiap *use case* digunakan *tool* yang bernama *activity diagram*. Berikut ini merupakan gambaran aktifitas pada setiap *use case* :



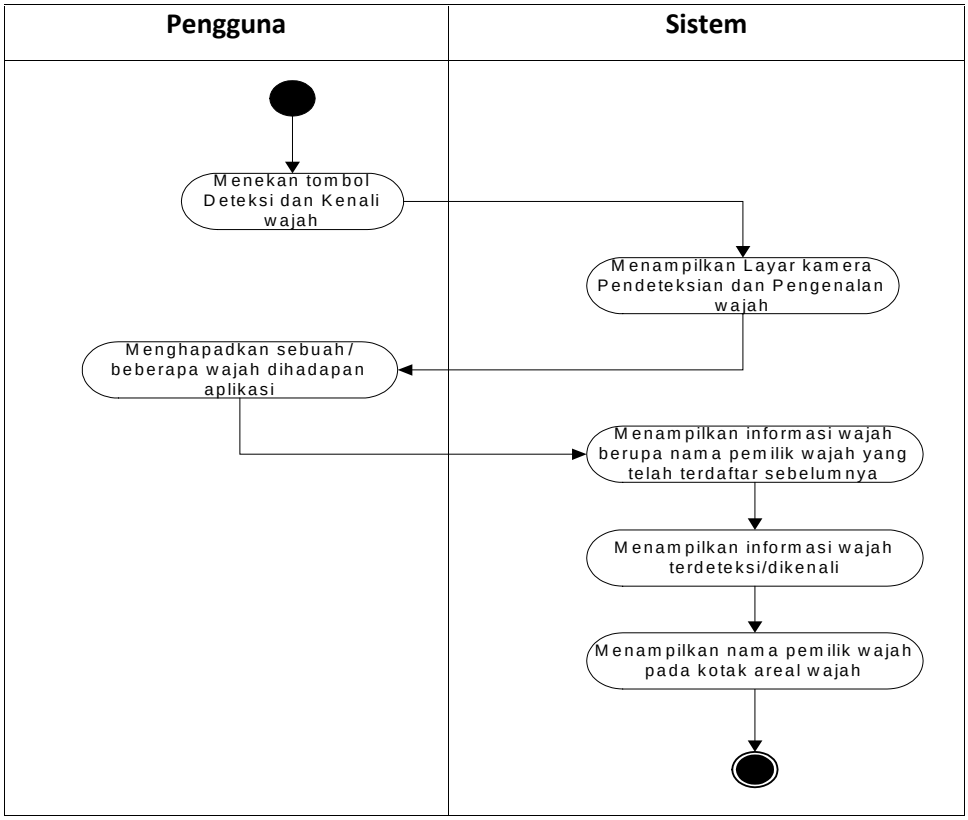
Gambar 4.15 *Activity Diagram Capture* Gambar



Gambar 4.16 *Activity Diagram* Deteksi Wajah



Gambar 4.17 *Activity Diagram* Pengenalan Wajah



Gambar 4.18 *Activity Diagram* Hasil Pendeteksian dan Pengenalan Wajah

4.2 Penggambaran Perancangan

Penerapan dari pengenalan wajah dalam penelitian ini akan digambarkan sebagai berikut:

4.2.1 Penggambaran *Multi-Face Detector* dengan Metode *Viola Jones*

Dalam penelitian ini proses pendeteksian wajah menggunakan sebuah file *xml* yang sudah ada pada *OpenCV* yaitu *Haar Cascade*. Terdapat beberapa parameter dalam file tersebut. Berikut adalah potongan program berdasarkan parameter yang digunakan dalam mendeteksi wajah:

```
CvHaarClassifierCascade       *cascade=0;
CvMemStorage                  *storage=0;

      IpImage* gray=img;

      If (!cascade) {
         Char* file =
          "C:/OpenCV2.0/data/haarcascade_frontalface_alt.xml";
         Cascade = (CvHaarClassifierCascade*) cvload(file, 0,0,0);
         Storage = cvCreateMemeStorage(0);
      }

      CvSeq* faces = cvHarDetectObjects(
         Gray,
         Cascade,
         Storage,
         1.1,
         3,
         CV_HAAR_DO_CANNY_PRUNING,
         cvSize(20, 20)
      );
```

Gambar 4.19 Potongan Kode Program *Cascade Classifier*

Berdasarkan potongan kode diatas terdapat beberapa fungsi yang menjadi parameter yang digunakan dalam penelitian ini yaitu: kode "img" yang merupakan *file* gambar asli, sedangkan kode "*gray*" merupakan *file* gambar yang akan dideteksi wajahnya. Kemudian kode "*cascade*" merupakan pilihan *file haar* untuk deteksi, bisa diganti selain wajah. Kode "*storage*" untuk tempat penyimpanan sementara. Kode "1.1" merupakan *Scale Increase Rate* untuk menspesifikasi seberapa cepat *OpenCV* meningkatkan skala untuk pendeteksian wajah setiap melewati bagian sebuah gambar. Semakin besar nilainya, semakin cepat proses pendeteksian, namun semakin besar juga kemungkinan adanya wajah

yang terlewatkan. Nilai standarnya adalah 1.1 (10%), namun dalam penelitian ini penulis menggunakan nilai yang lebih besar yaitu 1.2 (>10%), dengan pertimbangan menjaga *performance laptop* yang digunakan untuk implementasi aplikasi. Kemudian angka "3" merupakan *Minimum Neighbors Threshold*, digunakan sebagai pembatas, dan digunakan sebagai *cutoff level* untuk membuang atau menyimpan lokasi wajah yang terdeteksi. Angka 0 berfungsi menampilkan semua wajah yang terdeteksi oleh *Haar*. 3 berarti setiap tiga kotak, akan digabung menjadi satu kotak (*grouping*), 3 ini merupakan nilai *default*. Namun dalam penelitian ini penulis menggunakan nilai 10 dengan tujuan untuk menjaga kestabilan *performance* kinerja *laptop* yang lebih keras jika di-set dalam ukuran yang lebih kecil.

Fungsi *CV_HAAR_DO_PRUNING* adalah membuat *detector* hanya memproses bagian yang kemungkinan terdapat wajah. Jika kode *CV_HAAR_DO_PRUNING* diganti dengan 0, maka tidak akan dilakukan *canny pruning*.

Fungsi kode *cvSize(20,20)* merupakan ukuran bagian gambar yang akan diproses. Jika nilai ukuran ini diganti dengan ukuran lebih kecil dapat menyebabkan *CPU* bekerja lebih keras, namun jika ukuran lebih besar dapat mempengaruhi tingkat akurasi. Maka ukuran harus sesuai dengan resolusi gambar agar lebih efektif. Ukuran *default* dapat dilihat pada *file* xml dari Haar yang digunakan, di bagian *<size> 20 20 </size>*. Dalam penelitian ini penulis juga tetap menggunakan nilai *default* tersebut.

4.2.2 Penggambaran Pengenalan Wajah dengan Metode *Eigenface*

Dalam proses pengenalan wajah data *input* yang disimpan kedalam matrik dari hasil proses deteksi wajah akan dirata-ratakan nilainya dengan seluruh nilai matrik per *pixel.* Karena matrik ini menyimpan data citra wajah yang sudah terdeteksi dan dijadikan sebagai data *input*. Setiap citra ini selanjutnya akan diproyeksikan terhadap nilai rata-rata untuk mendapatkan perbedaan setiap citra kepada nilai rata-ratanya. Hasil dari pengurangan ini dapat disebut sebagai *mean subtracted image* berdimensi M x N^2. *mean subtracted image* akan menjadi data

input untuk menghitung *covariance matrix* dengan mengalikan masing-masing *mean subtracted image* dengan nilai *transpose*-nya sendiri. Dimensi dari *covariance matrix* ini akan sebesar M x N^2.

Menggunakan *C# covariance matrix* ini digunakan untuk menghitung nilai *eigen* dan *eigenvector.* Hasil dari nilai *eigen* dan *eigenvector* ini selanjutnya akan diurut dari terbesar hingga terkecil. Proses ini bertujuan untuk memperlihatkan nilai *eigenface* yang jelas menonjol dan lebih besar dari yang lain.

Proses selanjutnya mengalikan nilai *eigen* dan *eigenvector* dengan citra yang didapat dari proses *face detection* yang telah dikkurangi dengan rata-rata. Nilai hasil perkalian ini perlu dinormalisasikan agar tidak melebihi batas warna hitam yaitu 0, jika ada yang bernilai lebih kecil dari 0 akan di-set ke 0, sehingga diperoleh hasilnya nilai *eigenface*. Dengan mengalikan nilai *eigenface* dengan nilai citra yang didapat dari proses *face detection* yang sudah dikurangi dengan rata-rata kembali maka akan didapat nilai *eigenface*-nya. Nilai *eigenface* ini yang akan menjadi pembanding dengan nilai input baru untuk mendapatkan kecocokan dari objek yang dibandingkan.

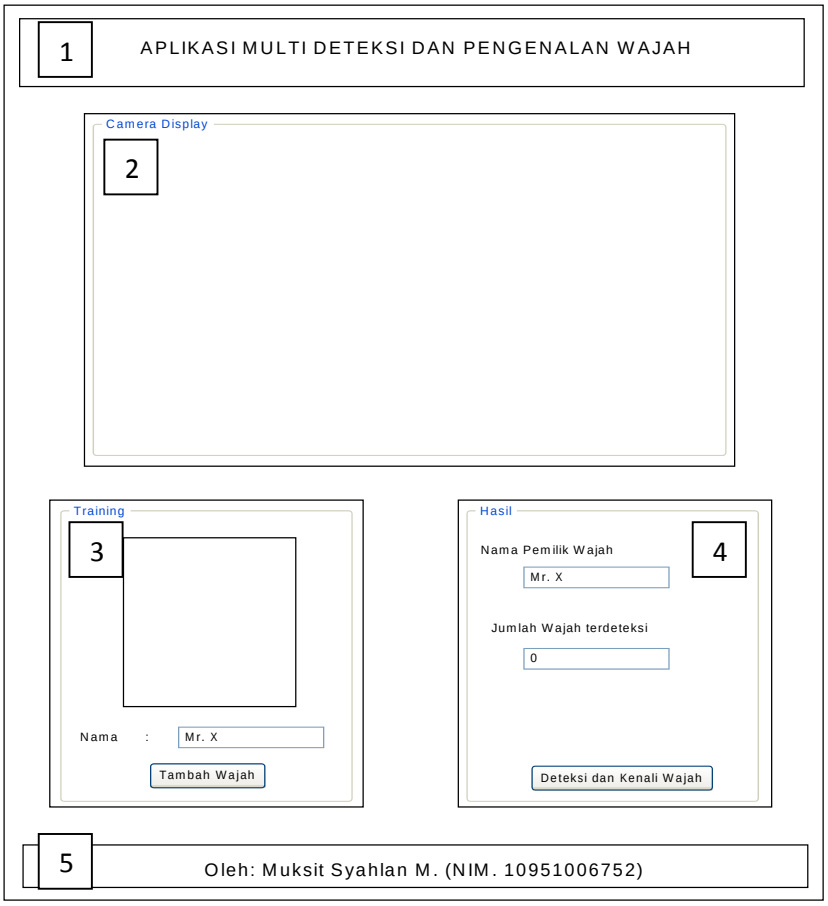
4.2.3 Perancangan *Interface*

Perancangan *interface* merupakan perancangan yang dibuat sebelum perangkat lunak dibuat. Perancangan *interface* yang digunakan dalam perangkat lunak ini adalah berbasis penampilan (*appereance based*). Perancangan *interface* aplikasi pendeteksi dan pengenalan wajah terdapat dua bagian, yaitu perancangan *form* aplikasi dan perancangan pesan peringatan. Perancangan ini dibuat menggunakan *Microsoft Office Visio 2007*.

4.2.3.1 Perancangan *Form* Aplikasi

Perancangan *form* model antar muka atau *interface* adalah bertujuan untuk mendesain gambaran sistem sebelum dibuat perancangan kedalam program. Dalam perancangan antar muka ini yang perlu diperhatikan adalah merancanag desain yang *user friendly* yang berfungsi untuk kenyamanan dalam pemakaian sistem oleh *user*, kemudian efisiensi dari penggunaan aplikasi agar hasilnya interaktif dan sesuai dengan yang diinginkan.

Untuk lebih jelasnya mengenai desain antar muka aplikasi ini, pada gambar 4.16 berikut ini ditampilkan model *interface form* utama aplikasi *multi-face detector* pada proses pengenalan wajah.



Gambar 4.20 *Interface Form* Utama

Keterangan:

Tabel 4.8 Keterangan *Interface* Sistem

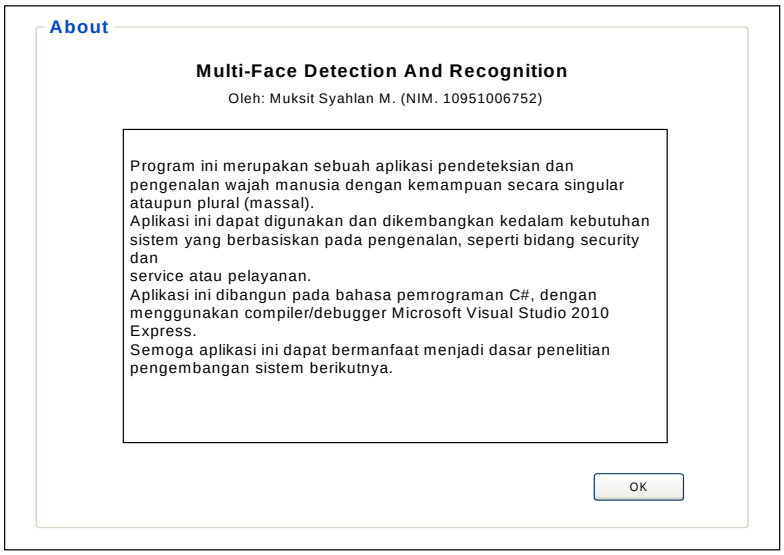
No.	Keterangan
1	Bagian *Header* Aplikasi yang berisi nama aplikasi
2	*ImageBox* yang merupakan bagian untuk menampilkan kamera secara *real time*
3	Merupakan proses training, berisi hasil cropping wajah yang telah di-*gray scale* dan di-*resize*, textfield untuk mengisi nama wajah dan ikon tambah wajah untuk menambahkan wajah kedalam *database*

Tabel 4.8 Keterangan *Interface* Sistem (Lanjutan)

No.	Keterangan
4	Merupakan tampilan hasil dari deteksi dan pengenalan wajah. *Textfield* pertama berisi nama wajah yang diasumsikan mirip oleh aplikasi, *textfield* kedua berisi jumlah wajah yang terdeteksi oleh aplikasi dan berisi tombol "deteksi dan kenali wajah" yang merupakan intruksi awal sebelum sistem bekerja
5	Merupakan bagian *footer* aplikasi yang berisi nama *developer* aplikasi

Disaat aplikasi baru di-*run* atau dieksekusi, maka akan tampil form seperti diatas. Namun aplikasi ini tidak lansung bekerja, maka pengguna harus menekan tombol "Deteksi dan Kenali Wajah". Sehingga bagian *image box* menampilkan layar hasil kamera. Jika tidak ada wajah terdeteksi oleh aplikasi, maka pada bagian hasil akan menampilkan *textfield* nama " ", yang artinya wajah tidak ada dan jumlah wajah terdeteksi "0". Jika fungsi kamera tidak berfungsi, maka pada kotak hasil pengenalan wajah diberikan informasi "Wajah Tidak terdeteksi". Jika ada wajah yang terdeteksi, maka pada bagian hasil akan lansung memberikan nama pemilik wajah jika dikenali dan memberikan jumlah wajah yang tereteksi.

Berikut ini adalah perancangan form informasi tentang aplikasi dan *developer* aplikasi.

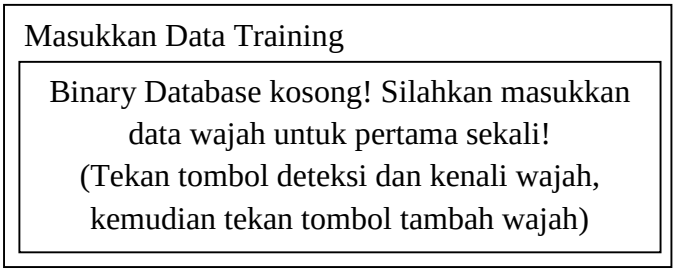


Gambar 4.21 *Interface Form* Tentang Aplikasi

4.2.3.2 Perancangan Pesan Peringatan

Perancangan pesan peringatan adalah respon yang diberikan aplikasi saat pengguna melakukan kesalahan atau keluar dari prosedur sistem yang telah ditetapkan pada aplikasi. Pesan peringatan ini berfungsi untuk memandu pengguna agar lebih mudah memahami cara kerja aplikasi. Berikut perancangan pesan peringatan aplikasi pendeteksi dan pengenalan wajah:

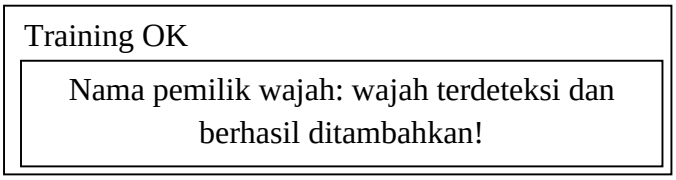
1. Pesan Peringatan “Masukkan Data Training”



Gambar 4.22 *Interface* Pesan Peringatan “Masukkan Data Training”

Keterangan: Pesan peringatan ini muncul saat aplikasi pertama sekali dijalankan, sehingga belum ada *database* wajah terisi yang berfungsi untuk proses pengenalan wajah sebagai data *training*.

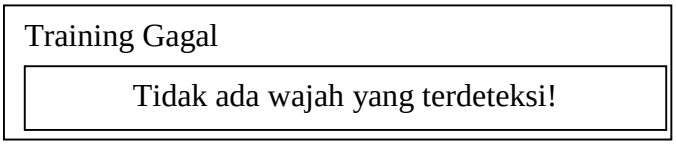
2. Pesan Peringatan “Training Ok”



Gambar 4.23 *Interface* Pesan Peringatan “Training Ok”

Keterangan: Pesan peringatan ini muncul saat pengguna melakukan aksi penekanan tombol “Tambah Wajah” setelah wajah terdeteksi.

3. Pesan Peringatan “Training Gagal”

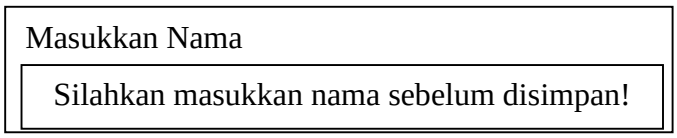


Gambar 4.24 *Interface* Pesan Peringatan “Training Gagal”

Keterangan: Pesan peringatan ini muncul saat pengguna melakukan aksi penekanan tombol “Tambah Wajah” namun pada layar kamera tidak

atau belum terdeteksi wajah. Wajah akan terdeteksi hingga muncul kotak merah areal wajah.

4. Pesan Peringatan “Masukkan Nama”



Gambar 4.25 *Interface* Pesan Peringatan “Masukkan Nama”

Keterangan: Pesan peringatan ini muncul saat pengguna melakukan aksi penekanan tombol “Tambah Wajah” namun *textfield* nama wajah tidak diisi atau kosong.

BAB V
IMPLEMENTASI dan PENGUJIAN

Setelah melakukan tahap pengembangan perangkat lunak pada bab analisa dan perancangan, maka dilanjutkan pada tahap pengembangan perangkat lunak implementasi dan pengujian.

5.1. Implementasi

Implementasi merupakan lanjutan dari tahap perancangan yaitu aplikasi siap dioperasikan pada keadaan sebenarnya, sehingga dapat diketahui apakah aplikasi yang dibuat telah menghasilkan tujuan yang diingan. Pada bagian ini akan diberikan gambaran mengenai implementasi perangkat lunak pengenalan wajah berdasarkan hasil perancangan yang telah dibuat pada bab sebelumnya. Pada bab ini meliputi batasan implementasi, lingkungan implementasi, serta implementasi antarmuka hasil perancangan.

5.1.1 Batasan Implementasi

Batasan untuk implemetasi perangkat lunak pendeteksi dan pengenalan wajah ini adalah sebagai berikut:

1. Ukuran citra wajah hasil *cropping* sebagai citra referensi dijadikan berukuran sama, yaitu sebesar 100 x 100 *pixel.*
2. Citra wajah yang dimasukkan kedalam *folder* penyimpanan hasil *cropping* untuk proses pengenalan wajah adalah berekstensi *.bmp.
3. Posisi wajah yang akan dideteksi ataupun dikenali adalah tegak dan menghadap kamera.
4. Dalam penelitian tidak menghitung kecepatan pendeteksian ataupun pengenalan wajah.
5. Wajah yang dideteksi berasal dari *live video* pada kamera secara *real time*, tidak dapat meng-*capture* gambar wajah dari file gambar atau *image.*

6. Tidak membahas pengukuran pengaruh faktor pencahayaan dan *motion* terhadap hasil pendeteksian dan pengenalan wajah.

5.1.2 Lingkungan Implementasi

Lingkungan implementasi dan pengujian sistem ada dua yaitu lingkungan perangkat keras dan lingkungan perangkat lunak.

5.1.2.1 Lingkungan Perangkat Keras

Perangkat keras yang digunakan dalam implementasi pada penelitian ini mempunyai spesifikasi sebagai berikut:

1. *Processor* : *AMD Dual-Core C*-50
2. *Monitor* : *Acer* 11,6”
3. *VGA* : *AMD Radeon HD* 6250 *Graphics*
4. *Memory* : 2 GB
5. *Camera* : *External Webcam Lexmark X422, resolution up to* 5 *megapixel*

5.1.2.2 Lingkungan Perangkat Lunak

Perangkat lunak pendeteksi dan pengenalan wajah ini dikembangkan dengan menggunakan beberapa perangkat lunak sebagai berikut:

1. Sistem Operasi : *Windows 7 Ultimate 32-bit Operating System*
2. Bahasa pemrograman : C# (*C Sharp*) dan *xml*
3. *Editing* kode program : *Microsoft Visual C# 2010 Express* dan *Text Editor Notepad ++*
4. *Compiler/Debugger* : *Microsoft Visul Studio 2010 Express*
5. Desain antarmuka : *Microsoft Office Visio 2007*

5.1.3 Implementasi Antarmuka

Rancangan antarmuka perangkat lunak pendeteksi dan pengenalan wajah ini diimplementasikan dengan menggunakan bahasa pemrograman C# dan *compiler Microsoft Visual Studio 2010 Express.* Berikut ini akan dijelaskan secara

umum seluruh *interface* yang terdapat pada perangkat lunak pendeteksi dan pengenalan wajah yang meliputi:

1. *Form* Utama Aplikasi Multi Deteksi dan Pengenalan Wajah

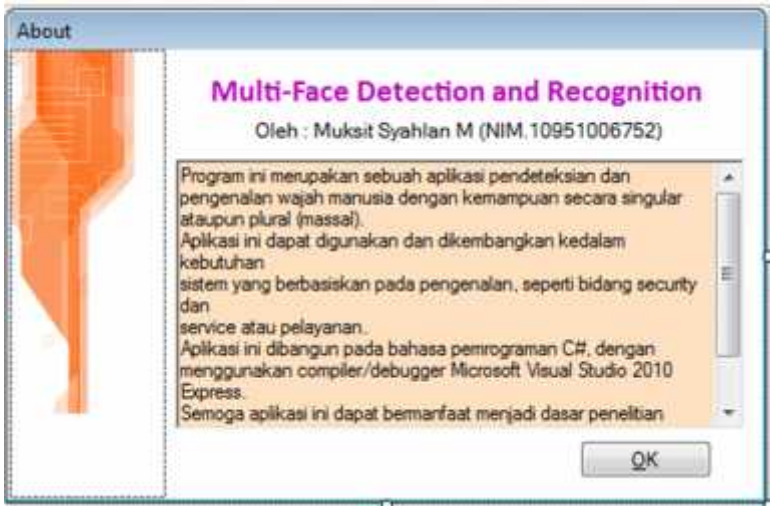
Gambar 5.1 *Form* Utama Aplikasi

Form utama ini merupakan tampilan awal aplikasi ketika di-*run.* Pada tampilan utama aplikasi ini terdapat 3 komponen utama, yaitu:

a. *Image Box Frame Grabber* yang berfungsi untuk menampilkan *live video* kamera untuk pendeteksian dan pengenalan wajah.
b. *Group Box Training* yang berfungsi sebagai memproses data training, yaitu berupa penambahan data *training* dari hasil *capture* dari kemera dan penamaan wajah.
c. *Group Box* Hasil berfungsi untuk menampilkan hasil dari pendeteksian dan pengenalan wajah, yaitu berupa nama pemilik wajah dan jumlah wajah terdeteksi. Kemudian didalamnya terdapat tombol ”deteksi dan kenali

wajah" yang berfungsi untuk mengaktifkan aplikasi dalam melakukan pendeteksian dan pengenalan wajah.

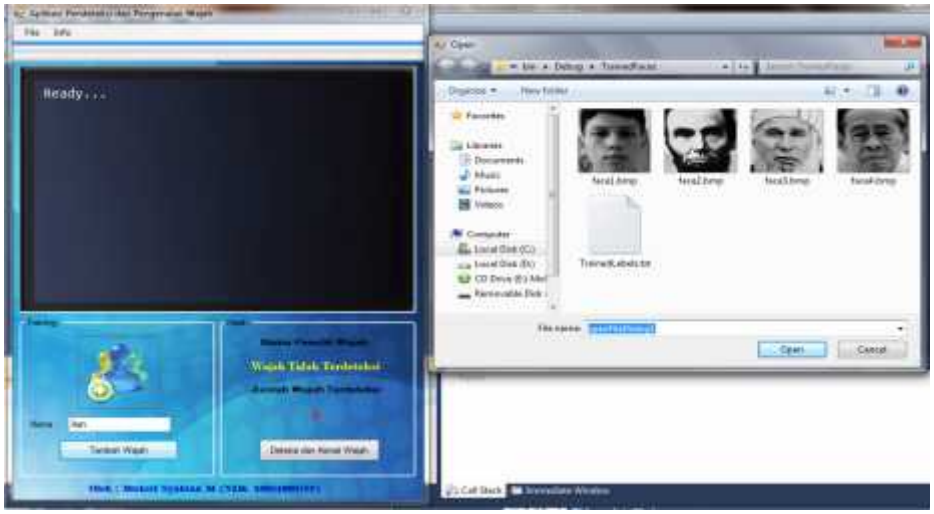
2. *Form* Informasi Tentang Aplikasi



Gambar 5.2 *Form About*

Form about ini berfungsi untuk memberikan informasi umum tentang perangkat lunak dan perangcangannya.

3. Tampilan Data Citra Wajah Referensi

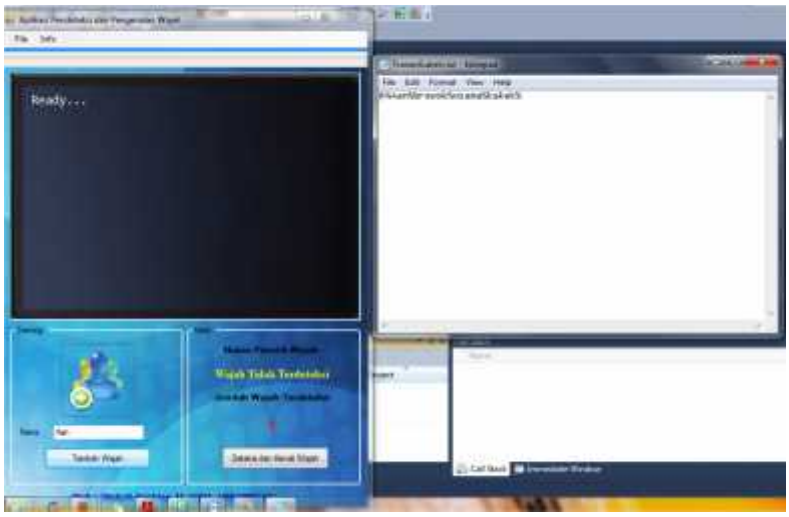


Gambar 5.3 Data Citra Wajah Referensi

Tampilan ini akan muncul ketika pengguna memilih menu File kemudian sub menu Data Wajah dan Gambar. Proses ini berfungsi untuk melihat hasil dari

penambahan sebuah gambar wajah referensi kedalam *folder* yang telah disediakan dan dapat juga untuk penghapusan atau pengeditan citra wajah referensi.

4. Tampilan Label Nama Citra Wajah Referensi



Gambar 5.4 Label Nama Citra Wajah Referensi

Tampilan ini akan muncul ketika pengguna memilih menu File kemudian sub menu Data Wajah dan Label. Proses ini berfungsi untuk pengeditan atau pembeharuan sebuah atau beberapa nama pemilik wajah referensi kedalam sebuah *file* yang ber-ekstensi **.txt* yang telah disediakan sebagai inisialisasi wajah dalam proses pengenalan wajah. Nama pemilik citra wajah ini diurut berdasarkan nomor urut citra wajah.

5.2. Pengujian

Tahap pengujian atau *testing* dilakukan setelah selesai tahap pembuatan dan seluruh data telah dimasukkan. Suatu hal yang tidak kalah penting dalam pegujian ini adalah aplikasi harus dapat berjalan dengan baik dilingkungan pengguna. Pengguna dapat merasakan manfaat serta kemudahan dari aplikasi dan dapat menggunakannya sendiri terutama untuk aplikasi interktif. Pada tahap pengujian, aplikasi diuji melalui pengujian terhadap citra dan pengujian melalui *blackbox*.

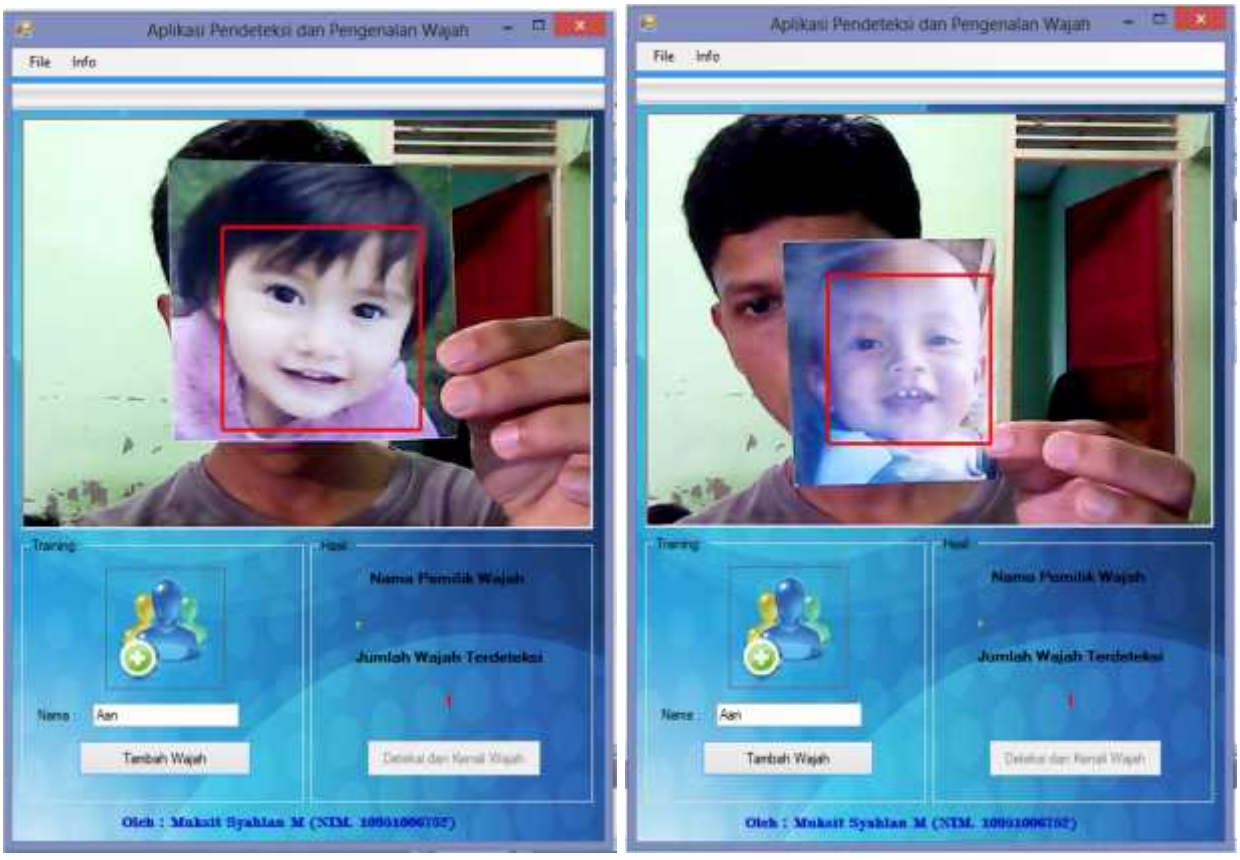
5.2.1 Pengujian Terhadap Citra

Pengujian perangkat lunak yang dilakukan dalam penelitian ini terdapat 2 kategori, yaitu: pengujian pendeteksian wajah dan pengujian pengenalan wajah. Berikut ini akan dijelaskan hasil pengujian masing-masing kategori.

5.2.1.1 Pengujian Pendeteksian Wajah

Dalam pengujian pendeteksian wajah ini penulis menggunakan beberapa parameter yang menjadi faktor diluar kebiasaan yang memungkinkan dapat mempengaruhi hasil pendeteksian wajah. Ini dilakukan agar dapat menguji tingkat akurasi dari aplikasi dalam mendeteksi wajah seseorang, yaitu:

1. Berdasarkan umur, terdiri dari pendeteksian wajah anak balita dan orang tua lansia. Untuk lebih jelasnya hasil pengujian ini dapat dilihat pada gambar 5.5 dan gambar 5.6 berikut ini.



Gambar 5.5 Pengujian Deteksi Wajah Anak Balita

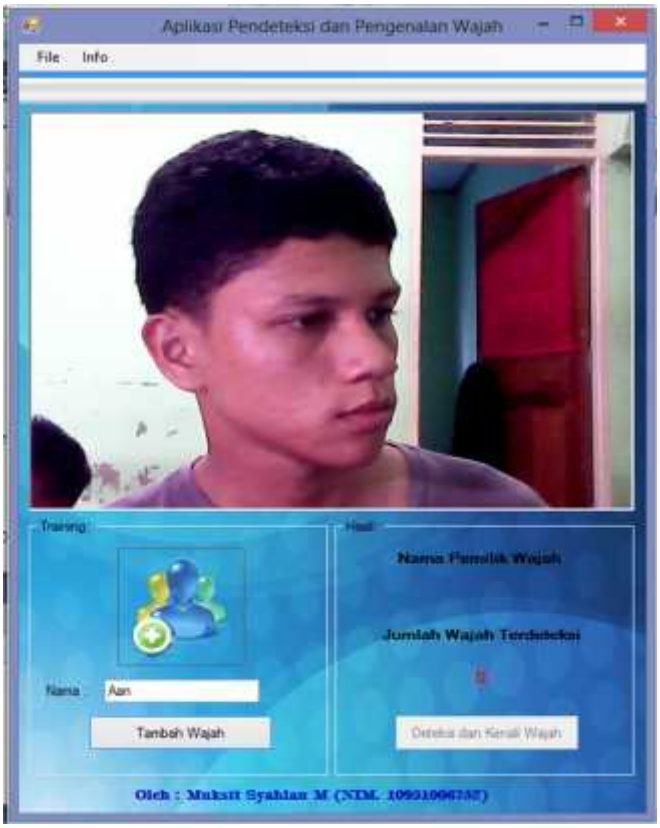
Hasil pendeteksian wajah ini menunjukkan bahwa aplikasi berhasil mendeteksi wajah anak balita yang memiliki umur antara 2-3 tahun.

Gambar 5.6 Pengujian Deteksi Wajah Orang Tua Lansia

Hasil pendeteksian wajah ini menunjukkan bahwa aplikasi berhasil mendeteksi wajah orang tua lansia yang umurnya berkisar lebih dari 60 tahun.

2. Berdasarkan gaya wajah, yaitu beberapa model wajah yang memungkinkan dapat mempengaruhi hasil dari pendeteksian wajah. Dalam penelitian ini penulis mengambil pengujian dengan beberapa model gaya wajah yaitu: gaya normal, menoleh dan memejamkan mata. Tiga model gaya wajah ini hanyalah sebagai sampel penelitian dari banyak kemungkinan yang dapat terjadi pada model gaya wajah. Untuk lebih jelasnya dapat dilihat pada gambar 5.7 sampai gambar 5.9 berikut ini.

Gambar 5.7 Pengujian Deteksi Wajah dengan Model Gaya Normal



Gambar 5.8 Pengujian Deteksi Wajah dengan Model Gaya Menoleh

Berdasarkan gambar hasil pengujian aplikasi diatas menunjukkan bahwa aplikasi berhasil mendeteksi wajah dengan gaya wajah dalam keadaan normal. Sedangkan wajah dengan model gaya menoleh, aplikasi tidak sanggup mendeteksi wajah tersebut, yang menunjukkan pendeteksian gagal dilakukan. Hal ini disebabkan karena *template* wajah yang dijadikan rujukan dalam *training* citra wajah dalam penelitian ini hanyalah pada posisi depan (*frontal face*), sedangkan untuk posisi miring ataupun menoleh yang cukup signifikan diperlukan model fitur baru yang mampu merotasikan citra sub-*window*, penelitian ini kemudian dikembangkan oleh Lienhart menggunakan 45^0 *Haar Like-Feature*. Selengkapnya dapat dilihat pada lampiran A.

Gambar 5.9 Pengujian Deteksi Wajah dengan Model Gaya Memejamkan Mata

Gambar hasil pengujian aplikasi diatas menunjukkan bahwa aplikasi masih berhasil mendeteksi wajah dengan model wajah dalam keadaan memejamkan mata.

3. Pendeteksian wajah dengan penambahan aksesoris dan perubahan fisik yang disebabkan oleh faktor usia seperi pertumbuhan kumis dan jenggot pada wajah. Dalam penelitian ini penulis memberikan aksesoris pada wajah berupa topi, kaca mata hitam dan masker. Untuk lebih jelasnya dapat dilihat pada gambar 5.10 sampai dengan gambar 5.15 berikut ini.

Gambar 5.10 Pengujian Deteksi Wajah dengan Penambahan Kumis

Berdasarkan gambar hasil pengujian aplikasi diatas menunjukkan bahwa aplikasi masih berhasil mendeteksi wajah dengan penambahan kumis. Selain itu pengujian juga dilakukan dengan penambahan jenggot pada wajah, dan hasilnya ternyata aplikasi masih mampu mendeteksi bagian wajah yang telah ditambahi jenggot. Untuk lebih jelasnya dapat dilihat pada gambar 5.11 berikut ini.

Selanjutnya adalah pengujian pendeteksian wajah yang telah ditambahkan aksesoris. Dalam pengujian yang dilakukan penelitian ini aksesoris yang ditambahkan pada wajah dibatasi hanya aksesoris topi, kaca mata hitam dan masker. Aksesoris yang digunakan ini hanya sebagai sampel pengujian meskipun pada kehidupan nyata banyek kemungkinan aksesoris lain yang dapat ditambahkan pada wajah. Hasil pengujian pendeteksian wajah dengan penembahan aksesoris dapat dilihat pada gambar 5.12 sampai 5.15 berikut ini.

Gambar 5.11 Pengujian Deteksi Wajah dengan Penambahan Jenggot

Gambar 5.12 Deteksi Wajah dengan Penambahan Topi

Berdasarkan gambar hasil pengujian aplikasi diatas, yaitu pendeteksian wajah yang telah ditambahkan aksesoris topi. Hasil pengujian menunjukkan bahwa aplikasi masih berhasil mendeteksi wajah dengan penambahan aksesoris topi pada wajah.

Gambar 5.13 Pengujian Deteksi Wajah dengan Penambahan Kaca Mata Hitam

Berdasarkan gambar hasil pengujian aplikasi diatas, yaitu pendeteksian wajah yang telah ditambahkan aksesoris kaca mata hitam. Hasil pengujian menunjukkan bahwa aplikasi masih berhasil mendeteksi wajah dengan penambahan aksesoris kaca mata hitam pada wajah.

Gambar 5.14 Pengujian Deteksi Wajah dengan Penambahan Masker Menutupi Hidung dan mulut

Gambar 5.15 Pengujian Deteksi Wajah dengan Penambahan Masker Sampai Mulut

Berdasarkan gambar hasil pengujian aplikasi diatas menunjukkan bahwa aplikasi tidak berhasil mendeteksi wajah dengan penambahan aksesoris masker pada wajah yang menutupi wajah hingga hidung. Dan juga penambahan aksesoris masker pada wajah yang hanya menutupi wajah pada bagian mulut saja, aplikasi juga tidak sanggup mendeteksi bagian wajah.

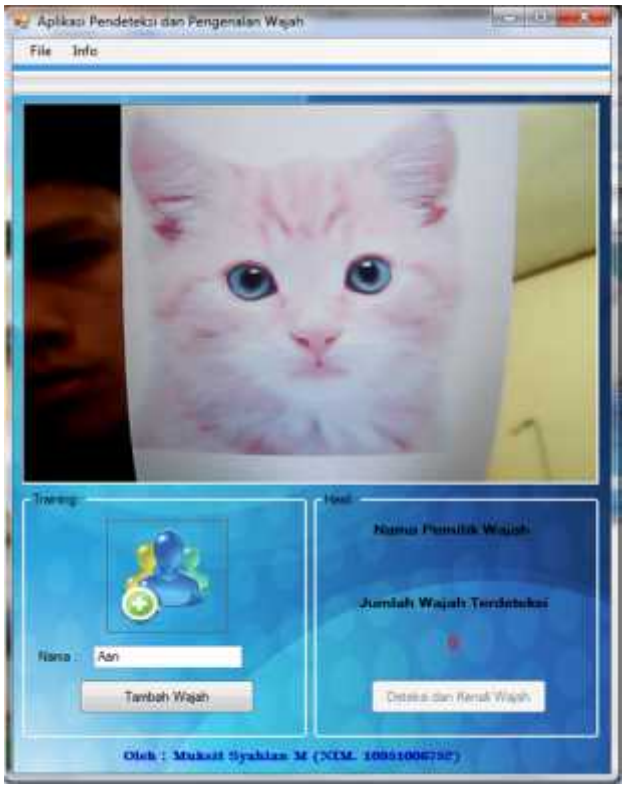
Hal yang paling mendasar pemakaian masker pada daerah wajah mngindikasikan proses deteksi wajah gagal dilakukan sistem disebabkan karena *library* pendeteksian wajah pada *OpenCV* yang digunakan dalam penelitian ini adalah file xml *frontal face*, file ini hanya berisi hasil *training* wajah secara utuh. Maka ketika citra wajah yang dideteksi tidak sempurna yang menghilangkan ciri utama wajah seperti bagian mulut atau hidung menjadikan sistem tidak mampu mendeteksi citra tersebut adalh wajah.

4. Pendeteksian wajah secara *plural* atau jamak (lebih dari 1 wajah). Dalam laporan penelitian ini penulis hanya menampilkan hasil pengujian pendeteksian wajah secara *plural* dengan memberikan 3 buah wajah yang setidaknya sudah mampu memmberikan gambaran bahwa aplikasi ini mampu melakukan pendeteksian wajah lebih dari satu orang. Hasil dari pengujian tersebut dapat dilihat pada gambar 5.14 berikut ini.

Gambar 5.16 Pengujian Deteksi Wajah Secara Plural

Berdasarkan gambar hasil pengujian aplikasi diatas, yaitu pendeteksian wajah yang secara *plural* atau jamak. Hasil pengujian menunjukkan bahwa aplikasi masih berhasil mendeteksi wajah secara *plural* atau jamak, dalam pengujian diatas terdeteksi 3 jumlah wajah yang terdiri dari seorang orang tua lansia dan dua orang anak balita.

5. Pengujian terhadap citra wajah binatang. Dalam penelitian ini binatang yang dijadikan objek untuk menguji apakah aplikasi mendeteksi wajah binatang adalah kucing, anjing dan orang utan. Hal ini dilakukan bertujuan untuk mengetahui akurasi pendeteksian wajah manusia dengan metode *Viola Jones*. Hasil pendeteksian ini dapat dilihat pada gambar 5.17 sampai dengan gambar 5.19 berikut ini.



Gambar 5.17 Pengujian Deteksi Wajah Kucing

Gambar 5.18 Pengujian Deteksi Wajah Anjing

Gambar 5.19 Pengujian Deteksi Wajah Orang Utan

Berdasarkan gambar hasil pengujian aplikasi diatas, yaitu pendeteksian wajah pada binatang. Hasil pengujian menunjukkan bahwa aplikasi sama sekali tidak mendeteksi wajah wajah binatang tersebut, meskipun dalam pengujian diatas terdeteksi dilakukan juga terhadap binatang yang secara bentuk wajahnya menyerupai wajah manusia yaitu orang utan. Hasil ini menyimpulkan bahwa binatang tidak dapat dideteksi wajahnya oleh aplikasi, hal ini disebabkan karena *library* yang digunakan tidak cocok dengan wajah binatang, karena wajah manusia memiliki perbedaan yang sangat mencolok pada ciri citra wajahnya dengan binatang, seperti pada dominasi kecerahan warna kulit, bentuk komponen wajah, dan tekstur wajah.

5.2.1.2 Pengujian Pengenalan Wajah

Proses pengenalan wajah merupakan proses akhir dari aplikasi pendeteksi dan pengenalan wajah dengan kemampuan *plural*. Pengujian pengenalan wajah ini dilakukan setelah proses pendeteksian dan *cropping* wajah berhasil dilakukan.

Sebelum dilakukan pengujian pengenalan wajah terhadap aplikasi terdapat sebuah tahap pra- *recognition*. Tahap ini merupakan proses persiapan data-data yang diperlukan sebelum pengenalan wajah, data-data tersebut yaitu: *trained face* dan label nama. Data *trained face* ini diperoleh dai hasil penambahan wajah dari *cropping image* wajah. Sedangkan label nama diperoleh dari *input* nama wajah sebelum proses *cropping image* wajah.

Dalam penelitian ini penulis telah menyiapkan sekitar 50 wajah referensi dengan pemilik dan kondisi yang berbeda-beda. Data wajah ini yang akan menjadi data pembanding antara citra wajah uji dengan citra wajah referensi pada proses pengenalan wajah. Untuk pengujian proses pengenalan wajah ini penulis menggunakan beberapa parameter yang menjadi faktor diluar kebiasaan yang memungkinkan dapat mempengaruhi hasil dari pengenalan wajah. Ini dilakukan agar dapat menguji tingkat akurasi dari aplikasi dalam pengenalan wajah, yaitu:

6. Berdasarkan umur, terdiri dari pengenalan wajah anak balita dan orang tua lansia. Untuk lebih jelasnya dapat dilihat pada gambar 5.17 dan gambar 5.18 berikut ini:

Gambar 5.17 Pengujian Pengenalan Wajah Anak Balita

Gambar 5.18 Pengujian Pengenalan Wajah Orang Tua Lansia

Hasil pengujian pengenalan wajah ini menunjukkan bahwa aplikasi berhasil mengenali dengan benar wajah anak balita yang memiliki umur antara 2-3 tahun dan orang tua lansia yang memiliki umur lebih dari 60 tahun.

7. Berdasarkan gaya wajah, yaitu beberapa model wajah yang dapat mempengaruhi hasil dari pengenalan wajah. Dalam penelitian ini penulis mengambil pengujian dengan beberapa model gaya wajah yaitu: gaya normal, memejamkan mata dan tertawa. Ketiga model gaya wajah ini hanyalah sebagai sampel penelitian dari banyak kemungkinan yang dapat terjadi pada model gaya wajah. Untuk lebih jelasnya dapat dilihat pada gambar 5.19 sampai gambar 5.21 berikut ini.

Gambar 5.19 Pengujian Pengenalan Wajah Kondisi Normal

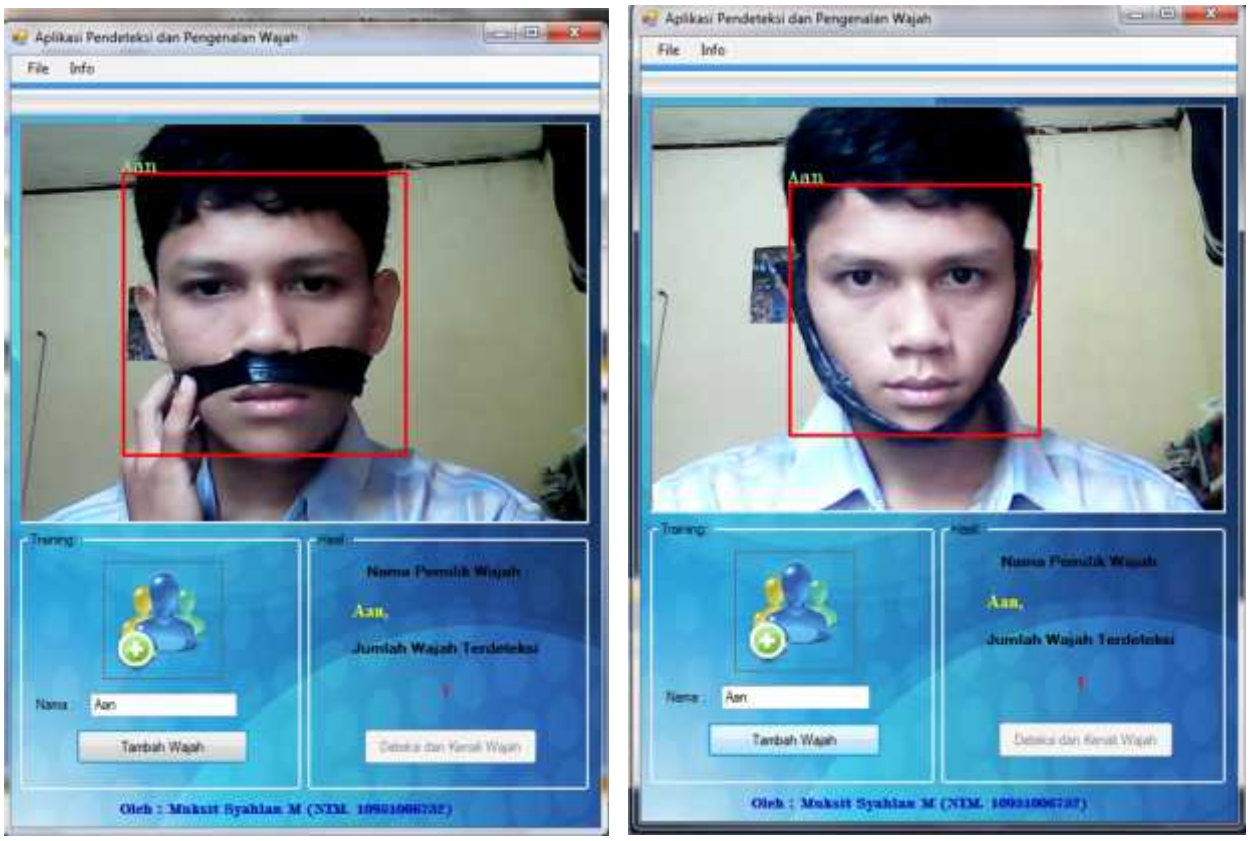
Hasil pengujian pengenalan wajah ini menunjukkan bahwa aplikasi berhasil mengenali dengan benar wajah pada kondisi normal.

Gambar 5.20 Pengujian Pengenalan Wajah Gaya Memejamkan Mata

Gambar 5.21 Pengujian Pengenalan Wajah Gaya Tertawa

Hasil pengujian pengenalan wajah dari kedua kondisi wajah seperti gambar diatas menunjukkan bahwa aplikasi berhasil mengenali dengan benar wajah pada kondisi memejamkan mata dan tertawa.

8. Pengenalan wajah dengan penambahan aksesoris dan perubahan fisik yang disebabkan oleh faktor usia seperi pertumbuhan kumis dan jenggot pada wajah. Dalam penelitian ini penulis memberikan aksesoris pada wajah berupa topi dan kaca mata hitam. Untuk lebih jelasnya dapat dilihat pada gambar 5.22 sampai dengan gambar 5.24 berikut ini.



Gambar 5.22 Pengujian Pengenalan Wajah dengan Penambahan Kumis dan Jenggot

Hasil pengujian pengenalan wajah dari kedua kondisi wajah seperti gambar diatas menunjukkan bahwa aplikasi berhasil mengenali dengan benar wajah dengan penambahan kumis atau jenggot.

Gambar 5.23 Pengujian Pengenalan Wajah dengan Penambahan Topi

Gambar 5.24 Pengujian Pengenalan Wajah dengan Kaca Mata Hitam

Hasil pengujian pengenalan wajah dari kedua kondisi wajah seperti gambar diatas menunjukkan bahwa aplikasi masih berhasil mengenali dengan benar wajah dengan penambahan aksesoris topi, namun aplikasi gagal mengenali wajah dengan benar ketika wajah diberi penambahan aksesoris kaca mata hitam.

9. Pengenalan wajah secara *plural* atau jamak (lebih dari 1 wajah). Dalam laporan penelitian ini penulis hanya menampilkan hasil pengujian pengenalan wajah secara *plural* dengan memberikan 3 buah wajah yang setidaknya sudah mampu memmberikan gambaran bahwa aplikasi ini mampu melakukan pendeteksian wajah lebih dari satu orang. Hasil dari pengujian tersebut dapat dilihat pada gambar 5.27 berikut ini.

Gambar 5.25 Pengujian Pengenalan Wajah *Plural*

Hasil pengujian pengenalan wajah ini menunjukkan bahwa aplikasi berhasil mengenali dengan benar tiga buah wajah anak kecil yang terdetsi kamera.

5.2.1.3 Evaluasi Hasil Pengujian Aplikasi

Program aplikasi diuji dengan melatih semua subyek yang ada didalam *folder image.* Subyek yang ikut dalam pengujian berjumlah 5 citra wajah, baik dalam pengujian pendeteksian maupun pengenalan wajah. Sedangkan jumlah total citra yang dijadikan sebagai citra referensi adalah sebayak 53 citra wajah. Untuk lebih jelasnya mengenai citra referensi yang digunakan dalam pengujian ini dapat dilihat pada gambar 5.26 dibawah ini.

Gambar 5.26 Citra Wajah Referensi

Tabel berikut merupakan hasil dari pendeteksian wajah dengan berbagai kondisi.

Tabel 5.1 Hasil Pengujian Pendeteksian Wajah

No	Deskripsi Wajah	Status Hasil
1	Wajah anak balita dengan umur 2-3 tahun	berhasil
2	Wajah orang tua lansia	berhasil
3	Wajah dengan gaya normal	berhasil
4	Wajah dengan gaya menoleh	gagal
5	Wajah dengan gaya memjamkan mata	berhasil
6	Wajah dengan penambahan kumis	berhasil
7	Wajah dengan penambahan jenggot	berhasil
8	Wajah dengan penambahan topi	berhasil

Tabel 5.1 Hasil Pengujian Pendeteksian Wajah (Lanjutan)

No	Deskripsi Wajah	Status Hasil
9	Wajah dengan penambahan kaca mata hitam	berhasil
10	Wajah dengan penambahan masker menutupi hidung dan mulut	gagal
11	Wajah dengan penambahan masker menutupi mulut	gagal
12	Wajah *plural* (jamak)	berhasil

Tabel berikut merupakan hasil dari pengenalan wajah dengan berbagai kondisi.

Tabel 5.2 Hasil Pengujian Pengenalan Wajah

No	Deskripsi Wajah	Status Hasil
1	Wajah anak balita dengan umur 2-3 tahun	berhasil
2	Wajah orang tua lansia	berhasil
3	Wajah dengan gaya normal	berhasil
5	Wajah dengan gaya tertawa	berhasil
6	Wajah dengan gaya memjamkan mata	berhasil
7	Wajah dengan penambahan kumis	berhasil
8	Wajah dengan penambahan jenggot	berhasil
9	Wajah dengan penambahan topi	berhasil
10	Wajah dengan penambahan kaca mata hitam	gagal
11	Wajah *plural* (jamak)	berhasil

Berdasarkan tabel 5.1 diatas, maka dapat dihitung tingkat akurasi dari pengujian pendeteksian wajah dan pengenalan wajah. Untuk tingkat akurasi pendeteksian wajah diperoleh berdasarkan kalkulasi berikut ini:

$$Hasil = \frac{jumlah\ wajah\ berhasil\ dideteksi}{total\ pengujian\ pendeteksian}\ x\ 100\%$$

$$Hasil = \frac{9}{12}\ x\ 100\%$$

$$Hasil = 75\%$$

Sedangkan berdasarkan tabel 5.2 diatas untuk tingkat akurasi pengenalan wajah diperoleh berdasarkan kalkulasi berikut ini:

$$Hasil = \frac{jumlah\ wajah\ berhasil\ dikenali}{total\ pengujian\ pengenalan}\ x\ 100\%$$

$$Hasil = \frac{10}{11}\ x\ 100\%$$

$$Hasil = 91\%$$

Dari hasil kalkulasi diatas diperoleh bahwa tingkat akurasi pendeteksian wajah berdasarkan pengujian yang penulis lakukan adalah sebesar 75%. Sedangkan tingkat akurasi pengenalan wajah adalah sebesar 91%.

5.2.2 Pengujian dengan Metode *Blackbox*

Pengujian dengan metode *blakbox* dilakukan dengan menguji setiap aksi yang dapat dilakukan pada aplikasi progam. Untuk lebih jelasnya dapat dilihat pada tabel 5.3 sampai 5.6 berikut ini.

Tabel 5.3 Pengujian Pendeteksian Wajah

Deskripsi	Aksi yang Diberikan	Hasil yang diharapkan	Hasil uji
Pengujian terhadap proses awal sebelum pendeteksian	Meng-klik tombol “Deteksi dan Kenali Wajah”	Ketika pengguna melakukan aksi terhadap aplikasi dengan meng-klik tombol “Deteksi dan Kenali Wajah”, maka aplikasi akan mengaktifkan layar *Image Box Frame Grabber* yang berfungsi untuk menampilkan *live video* dari *device* kamera untuk proses pendeteksian wajah.	Sukses
Pengujian terhadap kemampuan mendeteksi wajah ketika dihadapkan wajah manausia didepan kamera aplikasi	Meng-klik tombol “Deteksi dan Kenali Wajah”, kemudian hadapkan sebuah atau beberapa wajah didepan kamera	Aplikasi akan menampilkan hasil deteksi wajah pada layar *Image Box Frame Grabber* yang berfungsi untuk menampilkan *live video* dari *device* kamera dengan memberikan bingkai berwarna merah pada areal wajah yang berhasil terdeteksi oleh aplikasi	Suskses
Pengujian ketika aplikasi tidak mendeteksi adanya citra wajah dari *live video* dari *device* kamera	Meng-klik tombol “Deteksi dan Kenali Wajah”, kemudian hadapkan kamera aplikasi pada area yang tidak ada wajah	Aplikasi akan menampilkan pada kotak hasil jumlah wajah terdeteksi menunnjukkan angka 0 yang menandakan tidak ada wajah terdeteksi	Suskes

Tabel 5.4 Pengujian Penambahan Citra Wajah Referensi

Deskripsi	Aksi yang Diberikan	Hasil yang diharapkan	Hasil uji
Pengujian penambahan citra wajah untuk sebagai citra referensi dalam pengenalan wajah	Meng-klik tombol “Tambah Wajah”	Ketika pengguna melakukan aksi terhadap aplikasi dengan meng-klik tombol “Tambah wajah”, maka aplikasi akan menampilkan hasil *cropping* areal wajah dan dirubah dalam bentuk *grayscale* dengan ukuran standar 100 x 100 *pixel.* Kemudian aplikasi menampilkan pesan bahwa wajah berhasil ditambahkan kedalam *folder* citra wajah referensi yang disediakan dan nama pemilik wajah otomatis akan masuk kedalam file label nama yang berformat *.txt yang telah disediakan.	Sukses
Pengujian ketika aplikasi tidak mendeteksi adanya citra wajah dari *live video* dari *device* kamera	Meng-klik tombol “Tambah Wajah”, kemudian hadapkan kamera aplikasi pada area yang tidak ada wajah	Aplikasi akan menampilkan pesan peringatan “training gagal” bahwa tidak ada wajah terdeteksi	Suskes
Pengujian penambahan citra wajah untuk sebagai citra referensi dalam pengenalan wajah namun tidak memasukkan nama pemilik wajah	Meng-klik tombol “Tambah Wajah”	Aplikasi akan menampilkan pesan peringatan “Silahkan masukkan nama sebelum disimpan”	Suskses

Tabel 5.5 Pengujian Pengenalan Wajah

Deskripsi	Aksi yang Diberikan	Hasil yang diharapkan	Hasil uji
Pengujian terhadap proses awal sebelum pengenalan	Meng-klik tombol “Deteksi dan Kenali Wajah”	Ketika pengguna melakukan aksi terhadap aplikasi dengan meng-klik tombol “Deteksi dan Kenali Wajah”, maka aplikasi akan mengaktifkan layar *Image Box Frame Grabber* yang berfungsi untuk menampilkan *live video* dari *device* kamera untuk proses pengenalan wajah.	Sukses
Pengujian terhadap kemampuan mengenali wajah ketika dihadapkan wajah manausia didepan kamera aplikasi yang terdapat dalam citra referensi	Meng-klik tombol “Deteksi dan Kenali Wajah”, kemudian hadapkan sebuah atau beberapa wajah didepan kamera	Aplikasi akan menampilkan hasil deteksi wajah pada layar *Image Box Frame Grabber* yang berfungsi untuk menampilkan *live video* dari *device* kamera dengan memberikan bingkai berwarna merah pada areal wajah yang berhasil terdeteksi oleh aplikasi kemudian muncul label nama pemilik wajah diatas kotak bingkai tersebut sesuai pemilik wajah	Suskses
Pengujian terhadap kemampuan mengenali wajah ketika dihadapkan wajah manausia didepan kamera aplikasi yang tidak terdapat dalam citra referensi	Meng-klik tombol “Deteksi dan Kenali Wajah”, kemudian hadapkan sebuah atau beberapa wajah didepan kamera	Aplikasi akan menampilkan hasil deteksi wajah pada layar *Image Box Frame Grabber* yang berfungsi untuk menampilkan *live video* dari *device* kamera dengan memberikan bingkai berwarna merah pada areal wajah yang berhasil terdeteksi oleh aplikasi namun aplikasi tidak memunculkan label nama pemilik wajah yang menandakan wajah tidak dikenali	Suskses
Pengujian ketika aplikasi tidak mendeteksi adanya citra wajah dari *live video* dari *device* kamera untuk proses pengenalan wajah	Meng-klik tombol “Deteksi dan Kenali Wajah”, kemudian hadapkan kamera aplikasi pada area yang tidak ada wajah	Aplikasi akan menampilkan pada kotak hasil nama pemilik wajah menunnjukkan *texfield* dalam kondisi kosong yang menandakan tidak ada wajah yang dikenali atau aplikasi menunjukan tanda “,” ketika ada wajah terdeteksi tanpa memberikan label nama	Suskes

5.2.3 Kesimpulan Pengujian

Berdasarkan pengujian yang telah dilakukan, baik pengujian terhadap citra maupun pengujian yang dilakukan dengan metode *blackbox,* maka dapat diambil beberapa kesimpulan sebagai berikut:

1. Berdasarkan hasil pengujian pendeteksian dan pengenalan wajah dengan mengkombinasikan metode *Viola Jones* dalam pendeteksian wajah dan metode *eigenface* untuk proses pengenalan wajah secara *real time* maka dapat disimpulkan bahwa hasil pendeteksian dengan metode *Viola Jones* cukup akurat dengan memberikan nilai akurasi berdasarkan perhitungan dari pengujian yang dilakaukn sebesar 75%. Sedangkan hasil pengenalan wajah menggunakan metode *eigenface* menggunakan hasil pendeteksian wajah *Viola Jones* menunjukkan tingkat akurasi yang tinggi yaitu sebesar 91%. Hal ini membuktikan bahwa aplikasi pengenalan wajah dengan mengkombinasikan 2 metode *Viola Jones* dan *eigenface* bekerja sangat baik.
2. Ada beberapa hal yang mempengaruhi akurasi hasil dari pendeteksian wajah, yaitu wajah tidak mampu dideteksi oleh aplikasi ketika wajah dalam kondisi tidak pada posisi tegak dengan benar didepan kamera dan ketika bagian wajah seperti hidung dan mulut tertutupi menggunakan masker, hijab atau cadar. Sedangkan hasil dari pengenalan wajah dipengaruhi oleh faktor penambahan aksesoris, seperti penambahan kaca mata hitam terbukti gagal untuk dikenali karena aplikasi memberikan nama pemilik wajah yang salah.
3. Secara keseluruhan proses pendeteksian wajah berdasarkan metode *Viola Jones* dan pengenalan wajah berbasis metode *eigenface* dengan kemampuan secara *plural* atau jamak yang dijalankan oleh aplikasi berjalan dengan lancer dan sesuai dengan tujuan yang diharapkan.

BAB VI
KESIMPULAN dan SARAN

6.1. Kesimpulan

Berdasarkan rumusan masalah dalam penelitian ini, yaitu merancang dan membangun sebuah aplikasi pendeteksian wajah dengan kemampuan *plural* atau jamak (*face detection*) dengan menggunakan metode *Viola Jones* pada pengenalan wajah (*face recognition*). Kemudian dilakukan pemecahan solusinya dengan pendekatan sistem yaitu mulai dari perancangan sampai analisa dan pengujian aplikasi, maka dapat ditarik beberapa kesimpulan sebagai berikut:

1. Pendeteksian wajah menggunakan metode *Viola Jones* memiliki kemampuan dan *performance* yang sangat baik, hal ini terlihat berdasarkan pengujian yang dilakukan terhadap identifikasi wajah dalam kondisi yang memungkinkan dapat mempengaruhi hasil pendeteksian wajah, hasil tersebut menunjukkan rata-rata 75% aplikasi mampu mendeteksi wajah tersebut.
2. Penggabungan dua metode yaitu *Viola Jones* sebagai dasar dari pendeteksian wajah secara *plural* dan *Eigenface* yang diterapkan dalam sistem pengenalan multi-wajah secara *real time* memberikan hasil yang sangat baik. Hal ini ditunjukkan dengan tingkat pengenalan selama pengujian, yaitu dengan hasil rata-rata 91 %.
3. Faktor-faktor yang menyebabkan kegagalan dalam proses pendeteksian wajah ini bergantung pada kondisi citra wajah, yaitu posisi dan kesempurnaan kenampakan wajah. Sedangkan faktor-faktor yang mempengaruhi kegagalan aplikasi dalam pengenalan wajah bergantung pada kondisi wajah yang telah diberi penambahan aksesoris yang menutupi bagian utama wajah seperti kaca mata hitam, serta faktor pencahayaan.
4. Metode *Viola Jones* menggunakan sebuah *image processing library* yang berisi data *training* wajah manusia secara utuh, sehingga hanya dapat

diaplikasikan untuk pengenalan wajah manusia. Dalam pengujian telah diuji cobakan beberapa wajah binatang tidak terdeteksi oleh aplikasi.

5. Dalam penelitian ini diimplementasikan sebuah *library* untuk mendeteksi bagian wajah menggunakan sebuah file xml yaitu frontal_face.xml yang berfungsi sebagai classifier untuk mendeteksi wajah secara frontal. Toleransi kemiringan sekitar 30^0. Untuk mendeteksi wajah dengan kemiringan hingga 45^0 dibutuhkan fitur baru yaitu *45^0 Haar Like-Feature*, namun tidak diimplementasikan dalam penelitian ini.
6. Kinerja aplikasi pendeteksi dan pengenalan wajah ini dipengaruhi oleh spesifikasi laptop, menggunakan spesifikasi yang rendah akan menyebabkan "*Not Responding*" pada aplikasi atau *slow motion* pada hasil tampilan kamera. Spesifikasi yang disarankan adalah menggunakan *processor* dengan kecepatan diatas *2.0 Ghz*, dan kamera *webcam* dengan kemampuan resolusi hingga *5 MP*.

6.2. Saran

Beberapa saran yang bisa diberikan untuk penelitian selanjutnya antara lain :

1. Perlu dilakukan penelitian lanjutan untuk membuat sistem pendeteksian yang mampu meng-*capture* wajah dengan posisi menoleh yang cukup signifikan yaitu sekitar 45^0.
2. Perlu dilakukan penelitian lebih lanjut untuk mengatasi kegagalan aplikasi dalam mendeteksi wajah pada kondisi miring lebih dari 45^0, hal ini dapat diatasi dengan menerapkan 45^0 *Haar Like-Feature* yang memiliki fungsi *rotated integral image*.
3. Perlu dilakukan penelitian lebih lanjut untuk mengukur pengaruh *brightness* atau pencahayaan terhadap hasil pendeteksian maupun pengenalan wajah berdasarkan penelitian ini.
4. Perlu dilakukan penelitian lanjutan untuk mendeteksi wajah seseorang meskipun dalam kondisi menggunakan masker ataupun cadar, ini dapat dilakukan dengan mengintegrasikan *library* pendeteksian objek wajah

dengan objek mata atau telinga untuk hasil pendeteksian wajah lebih yang akurat.

DAFTAR PUSTAKA

Dorman, Scott, "*Teach Yourself Visual C# 2010 in 24 Hours*", United States on America, Sams Publishing, USA, 2010.

Garilbaldy, Mukti W., "*Implementasi Algoritma Fractal Neighbour Distance untuk Face Recognition*" , Makalah Skripsi Institut Teknologi Bandung, Bandung, 2006.

Hewitt, Robin, "*Finding Face in Images*" SERVO Magazine, 2007.

Kurniawan, A., "*Aplikasi Absensi Kuliah Berbasis Identifikasi Wajah Menggunakan Metode Gabor Wavelet*" Makalah Skripsi Institut Teknologi Sepuluh Nopember (ITS) Surabaya, Surabaya, 2009.

Liu, J.G., Mason, P.J., "*Essential Image Processing and GIS for Remote Sensing*", John Wiley and Sons, Chichester, 2009.

Nathaneal, Ryan. *Face Recognition Menggunakan eigenface*. Makalah Skripsi, Universitas Kristen Petra, Surabaya, 2012.

Viola, Paul., Jones, Michael. "*Rapid Object Detection using a Boosted Cascade of Simple Features*" IEEE CVPR, 2001.

Prasetyo, Eri. "*Face Recognition System Design with Expression Position and Variation Method Using Eigenface*" Makalah Skripsi Universitas Gunadarma, 2009.

Referensi Manual, Visual Control 0.1, 2011.

Seo, N. *Tutorial: OpenCV haartraining (Rapid Object Detection With A Cascade of Boosted Classifiers Based on Haar-like Features)*, PukiWiki Plus, 2007.

Sholeh, Alfin, "*Pengembangan Sistem Pengenalan Wajah 2D dengan Implementasi Algoritma Eigenface dan Mahattan Distance*", Makalah Skripsi Universitas Pendidikan Indonesia, Bandung, 2013.

Turk, M., Pertland, A., "*Face Recognition using Eigenfaces, Conference on Computer Vision and Pattern Recognition*", Maui, HI , USA, pp. 586 – 591, 3 – 6 June 1991.

Sumber Internet :

Internet: Hewwit, Robin. April 2007 . *Face Recognition with Eigenface* , (Online), (http://www.cognotics.com/opencv/servo_2007_series/part_4/index.html, diakses 22 Februari 2013).

Internet: Emami, Shervin. 5 Januari 2012 . *Principal component Analysis* , (Online), (http://opencv.willowgarage.com/wiki/FaceRecognition, diakses 9 Maret 2013).

Internet: Teknomo, Kardi. 2006 *"Similarity Measurement"*, Copyrighted tutorials. (Online), (http:\\people.revoledu.com\kardi\tutorial\similarity\), diakses 11 Maret 2013.